# ABSTRACT

Title of dissertation: MULTIMODAL BIOMEDICAL DATA VISUALIZATION: ENHANCING NETWORK, CLINICAL, AND IMAGE DATA DEPICTION

Hsueh-Chien Cheng, Doctor of Philosophy, 2017

Dissertation directed by: Professor Amitabh Varshney
Department of Computer Science

In this dissertation, we present visual analytics tools for several biomedical applications. Our research spans three types of biomedical data: reaction networks, longitudinal multidimensional clinical data, and biomedical images. For each data type, we present intuitive visual representations and efficient data exploration methods to facilitate visual knowledge discovery.

Rule-based simulation has been used for studying complex protein interactions. In a rule-based model, the relationships of interacting proteins can be represented as a network. Nevertheless, understanding and validating the intended behaviors in large network models are ineffective and error prone. We have developed a tool that first shows a network overview with concise visual representations and then shows relevant rule-specific details on demand. This strategy significantly improves visualization comprehensibility and disentangles the complex protein-protein relationships by showing them selectively alongside the global context of the network.

Next, we present a tool for analyzing longitudinal multidimensional clinical datasets, that we developed for understanding Parkinson's disease progression. Detecting patterns involving multiple time-varying variables is especially challenging for clinical data. Conventional computational techniques, such as cluster analysis and dimension reduction, do not always generate interpretable, actionable results. Using our tool, users can select and compare patient subgroups by filtering patients with multiple symptoms simultaneously and interactively.

Unlike conventional visualizations that use local features, many targets in biomedical images are characterized by high-level features. We present our research characterizing such high-level features through multiscale texture segmentation and deep-learning strategies. First, we present an efficient hierarchical texture segmentation approach that scales up well to gigapixel images to colorize electron microscopy (EM) images. This enhances visual comprehensibility of gigapixel EM images across a wide range of scales. Second, we use convolutional neural networks (CNNs) to automatically derive high-level features that distinguish cell states in live-cell imagery and voxel types in 3D EM volumes. In addition, we present a CNN-based 3D segmentation method for biomedical volume datasets with limited training samples. We use factorized convolutions and feature-level augmentations to improve model generalization and avoid overfitting.

# VISUALIZATION OF COMPLEX BIOMEDICAL DATA : NETWORK, CLINICAL, AND IMAGERY DATA

by

Hsueh-Chien Cheng

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

## Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Amitabh Varshney, for his support and insights. It has been a pleasure and honor to work with him. I could not have imagined having a better advisor and mentor for my Ph.D. studies.

I would also like to thank Dr. Joseph JaJa, Dr. Dana Nau, Dr. Matthias Zwicker, and Dr. Antonio Cardone for their scholarly guidance towards the completion of my dissertation.

I am grateful to my collaborators for offering their expertise: Dr. Martin Meier-Schellersheim, Dr. Bastian R. Angermann, and Dr. Fengkai Zhang for designing rule-based protein interaction models; Dr. Lisa M. Shulman, Dr. Rainer von Coelln, Dr. Ann L. Gruber-Baldini for clinical reasoning with patient data; Dr. Alan Faden, and Dr. Bogdan Stoica for visualizing live-cell imagery; and Dr. Kedar Narayan and Dr. Sriram Subramanian for acquiring and analyzing microscopy images. I appreciate their invaluable feedback and contributions to my Ph.D. studies.

I would like to say a big thank you to everyone in the Graphics and Visual Informatics Laboratory: Horace, Sujal, Eric Krokos, Ruofei, Xuetong, Xiaoxu, Shuo, Tara, Eric Lee, and Sida.

Finally, I would like to thank my family for encouraging me to pursue my passion and career. Their love and support have been the best gift I will ever receive.

# Table of Contents

Acknowledgements ii

Table of Contents iii

List of Tables vi

List of Figures vii

1 Introduction 1
- 1.1 Visualization of Rule-based Reaction Networks . . . 1
- 1.2 Visualization of Temporal Changes in High-dimensional Clinical Data 3
- 1.3 Biomedical Image Understanding . . . 6
  - 1.3.1 Visualization of Microstructures in Gigapixel Microscopy Images 6
  - 1.3.2 Visualization of Live-cell Imagery . . . 8
  - 1.3.3 Deep-learning-assisted Volume Visualization . . . 10
  - 1.3.4 CNN-based Segmentation of Volumetric Microscopy Images . 12

2 Visualization of Rule-based Reaction Networks 15
- 2.1 Introduction . . . 15
- 2.2 Related Work . . . 17
- 2.3 Simmune Framework . . . 19
  - 2.3.1 Visual Representation of Reaction Rules . . . 20
- 2.4 Visual Analytics Tool: Simmune NetworkViewer . . . 24
  - 2.4.1 Network Graph . . . 24
  - 2.4.2 Visual Representation of Network Graph . . . 26
  - 2.4.3 User Interactions . . . 32
  - 2.4.4 Implementation . . . 36
- 2.5 Case Study . . . 37
- 2.6 Conclusions and Future Work . . . 42

3 Visualization of Temporal Changes in High-dimensional Clinical Data 44
- 3.1 Introduction . . . 44
- 3.2 Related Work . . . 48
- 3.3 Preliminary Study . . . 51
- 3.4 Data . . . 55
- 3.5 Visual Analytics Tool: Winnow . . . 56

3.5.1 Outcomes Panel . . . 58
3.5.2 Demographics Panel . . . 60
3.5.3 Analytics Panel . . . 62
3.5.4 Implementation . . . 63
3.6 Case Study . . . 63
3.6.1 Questions . . . 64
3.6.2 Discussion . . . 68
3.7 Conclusions and Future Work . . . 69

4 Overview of Biomedical Images and Convolutional Neural Networks 71
4.1 Introduction . . . 71
4.2 Background . . . 72
4.2.1 Biomedical Images . . . 72
4.2.2 Convolutional Neural Networks (CNNs) . . . 73

5 Visualization of Microstructures in Gigapixel Microscopy Images 79
5.1 Introduction . . . 79
5.2 Related Work . . . 83
5.2.1 Texture Analysis . . . 83
5.2.2 Image Segmentation . . . 85
5.3 Joint Intensity-texture Histogram . . . 87
5.4 Graph-based Multiscale Segmentation . . . 89
5.5 Hierarchical Microstructures Exploration . . . 92
5.6 Experiments . . . 93
5.6.1 Dataset 1: Zebrafish . . . 94
5.6.2 Dataset 2: Armadillidiidae . . . 98
5.6.3 Processing Time . . . 99
5.6.4 Discussion . . . 99
5.7 Conclusions and Future Work . . . 100

6 Visualization of Live-cell Imagery 102
6.1 Introduction . . . 102
6.2 Related Work . . . 105
6.2.1 Convolutional Neural Networks . . . 105
6.2.2 3D Convolutional Neural Networks . . . 106
6.2.3 Data . . . 107
6.3 CNN-based Feature Extraction . . . 107
6.4 User-mediated Color Annotation . . . 110
6.4.1 Color-based Annotation . . . 110
6.4.2 User-mediated Color Assignment . . . 111
6.4.3 Similarity-based Feature Reordering . . . 113
6.4.4 Vector Quantization . . . 114
6.5 Cell-state Trajectory Visualization . . . 115
6.6 Results and Discussion . . . 117
6.6.1 Results . . . 118

6.6.2 Discussion . . . . . . . . . . 118
6.7 Conclusions and Future Work . . . . . . . . . . 121

7 Deep-learning-assisted Volume Visualization 123
7.1 Introduction . . . . . . . . . . 123
7.2 Related Work . . . . . . . . . . 125
7.2.1 Volume Visualization . . . . . . . . . . 125
7.3 Motivation . . . . . . . . . . 127
7.3.1 Conventional Volume Visualization Design . . . . . . . . . . 129
7.3.2 Deep-learning-assisted Visualization Design . . . . . . . . . . 133
7.4 CNN-based Feature Extraction . . . . . . . . . . 135
7.5 Marching-cubes-based Visualization . . . . . . . . . . 137
7.6 User-mediated Voxel Classification . . . . . . . . . . 140
7.6.1 Similarity-based Feature Reordering . . . . . . . . . . 140
7.6.2 Vector Quantization . . . . . . . . . . 143
7.6.3 Hierarchical Volume Exploration . . . . . . . . . . 145
7.7 Results and Discussion . . . . . . . . . . 146
7.7.1 Segmentation Performance Measure . . . . . . . . . . 146
7.7.2 Dataset 1: Bacteria . . . . . . . . . . 147
7.7.3 Dataset 2: Hippocampus . . . . . . . . . . 153
7.7.4 Dataset 3: Multimodal Brain Tumor Image Segmentation Benchmark (BRATS) . . . . . . . . . . 156
7.7.5 Discussion . . . . . . . . . . 158
7.7.6 Rendering Speed and Computation Time . . . . . . . . . . 160
7.8 Conclusions and Future Work . . . . . . . . . . 161

8 CNN-based Segmentation of Volumetric Microscopy Images 164
8.1 Introduction . . . . . . . . . . 164
8.2 Related Work . . . . . . . . . . 166
8.2.1 CNN-based Volume Segmentation . . . . . . . . . . 166
8.2.2 Factorized Convolutions . . . . . . . . . . 167
8.2.3 Augmentations . . . . . . . . . . 168
8.3 Network Architecture . . . . . . . . . . 169
8.4 Factorized Convolution . . . . . . . . . . 172
8.5 Feature-level Augmentation . . . . . . . . . . 173
8.6 Jaccard Index-based Loss . . . . . . . . . . 176
8.7 Results and Discussion . . . . . . . . . . 177
8.8 Conclusions and Future Work . . . . . . . . . . 181

9 Conclusions and Future Work 182

List of Publications 185

Bibliography 187

# List of Tables

1.1 Segmentation results obtained by U-Net and eight network variants (A–H). The number of parameters is in millions. . . . . . . . . . . . . 13

2.1 The icons used for different molecule component states and binding site statuses. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

3.1 List of the 16 outcome measures and six domains of PD in our study. 57
3.2 Outcome measures that are significantly different ($p < 0.05$) by gender at year three (Y3) . . . . . . . . . . . . . . . . . . . . . . . . . 67

5.1 The images used in the experiment. Sizes are measured in gigapixels. 93
5.2 The running time (m) of four most time-consuming processes and the total running time of the proposed method. . . . . . . . . . . . . . 99

7.1 The architecture of the CNN . . . . . . . . . . . . . . . . . . . . . . . 136
7.2 The quality measures of the segmentation results for the bacteria and hippocampus datasets. Numbers separated by slashes correspond to different postprocessing techniques (none / class prior / fitted transformation). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150

8.1 Segmentation results obtained by U-Net and eight network variants (A–H). The number of parameters is in millions. . . . . . . . . . . . . 178

# List of Figures

1.1 The circular layout of a network put `IL4` at the center and other nodes from the center to the peripheral with respect to their distances to `IL4`. 2

1.2 Our visual analytics tool, Winnow, consists of three panels: the Outcomes panel (left), the Demographics panel (top right), and the analytics panel (bottom right). The colors represent the gender of patients (magenta for females and blue for males). . . . . . . . . . . . . 5

1.3 With our visualization tool, users can change the color highlights by traversing the segment hierarchy; adjacent segments are assigned distinguishable colors such that they are discernible at coarse resolutions. 7

1.4 The change in color (from green to red) of cell 39 from frame 49 to frame 54 suggests that the cell is dying. . . . . . . . . . . . . . . . . . 9

1.5 (a) Before spectral ordering, the similarity matrix, in which a bright pixel represents a pair of highly correlated features, does not show apparent pattern. After spectral ordering, highly correlated features are closer to each other. (b) The arbitrary order of features does not exploit the correlation among features, thus leading to broken surfaces. The spectrally ordered features allow creating a visualization that reveals the cytoplasm of the cell. . . . . . . . . . . . . . . . . . . . 11

1.6 The segmentation generated by model H, which uses 3D convolutions, is comparable to the ground truth; the segmentations generated by model D and U-Net [1], both use 2D convolutions only, misclassify the non-mitochondria region near the top left corner as mitochondria. 14

2.1 (a) A complex association where the two complexes on the left-hand side, `Ligand` and `Rec inactive`, bind and produce a `Ligated Receptor` complex. (b) A complex dissociation where `Ligated Receptor` on the left-hand side splits into a `Ligand` and a `Rec inactive` complex. (c) A complex transformation where the reacting complex `LigReg_Gab_GDP` transforms into the product complex `LigRec_Gabg_GTP`. 23

2.2 Overview of a G-protein network with 19 nodes and 29 edges created with a model that consists of 8 complex species and 11 complex reactions. Complex species nodes are displayed with the iconographic representation used in the Simmune framework. Intermediate nodes are displayed as small arrows indicating direction of reactions. . . . . 25

2.3 After specifying the reference species `IL4`, the network can be reorganized using level-based and circular layout. (a) The corresponding reference node `IL4` is placed at the top level; the other nodes are arranged with respect to their distances to the reference node. (b) The corresponding reference node `IL4` is placed at the center; the other nodes are arranged with respect to their distances to the reference node. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
2.4 (a) With the search term "gdp", the matching complex `LigRec_Gab_GDP` is highlighted in yellow. The non-matching complex species `LigRec_Gabg` has a matching complex and is therefore colored in light blue. (b) The complex species and complexes that contain the molecule `Ligand` are highlighted in yellow. . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
2.5 (a) The three complex species nodes show the three involved complexes, `Ligand`, `Rec inactive` and `Ligated Receptor`, after the complex association `receptor ligation` (shown in Figure 2.1a) is selected. (b) A hovering frame shows the two involved complexes, `LigRec_Gab_GDP` and `LicRec_Gabg_GTP`, after the complex transformation `Receptor mediated Galpha GDP GTP exchange` (shown in Figure 2.1c) is selected. . . . . . . . . . . . . . . . . . . . . . . . . . 34
2.6 After selecting the complex `Rec inactive`, the related edges generated by the two reaction rules in Figure 2.1a and Figure 2.1b are highlighted. Because other edges in the network remain unchanged, the visualization shows that `Rec inactive` participates in the two aforementioned reactions only. . . . . . . . . . . . . . . . . . . . . . . . . 35
2.7 A search is performed on the complex species `Receptor` to find complexes with an "off" state in the intracellular molecule component. The edges are highlighted to show that three reactions contain a reactant or product complex with an "off" state in the intracellular molecule component. . . . . . . . . . . . . . . . . . . . . . . . . . . 36
2.8 The complex species `EGFR` has five binding sites, four of them (e.g. with indices $1-4$) can be used to bind adaptors. Three molecule component states, "bndPLCg992", "bndPLCg1173" and "bndSHC1148" accommodate the binding constraints reported in [2], which are described as rules defining which adaptors can bind simultaneously to the EGFR. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
2.9 (a) Two states "bndPLCg992" and "bndPLCg1173", represented as red and blue squares, have to be "off" for PLC$\gamma$1 to be able to bind to site pY992. (b) All three states "bndPLCg992", "bndPLCg1173" and "bndSHC1148", represented as red, blue and green squares, have to be "off" for PLC$\gamma$1 to be able to bind to site pY1173. . . . . . . . 39
2.10 The network contains 17 nodes and 18 edges created from 11 complex species and 6 complex reactions. Here we select the complex association rule described in Figure 2.9a. . . . . . . . . . . . . . . . 40

2.11 (a) After specifying a search constraint where the state "bndPLCg992" is on, the visualization shows that PLC$\gamma$1 cannot bind to site pY992/pY1173. (b) After specifying a search constraint where the state "bndSHC1148" is on, the visualization shows that PLC$\gamma$1 cannot bind to site pY1173. 41

3.1 This screenshot of Winnow shows the results based on the Parkinson's Progression Markers Initiative (PPMI) dataset (Section 3.4). Winnow consists of three panels: the Outcomes panel (left), the Demographics panel (top right), and the analytics panel (bottom right). Here 55 patients (out of 320) with low baseline REM sleep behavior disorder (RBDQ $< 5$) and high RBDQ (RBDQ $\geq 5$) at the third annual visit are selected. The colors represent the gender of patients (magenta for females and blue for males). . . . . . . . . . . . . . . . . . 47
3.2 (a) Our early attempt created a hierarchy of clusters to allow splitting and merging of clusters following a pre-generated hierarchy. Nevertheless, the parallel set visualization of two sibling clusters: cluster #3 (figure (b)) and cluster #12 (figure (c)), are difficult to interpret even with three dimensions. . . . . . . . . . . . . . . . . . . . . . . . . . . 53
3.3 After using the bottom histogram for selecting patients with the most rapid progression in the motor domain ($\Delta$(UPDRS3) $> 12$), the corresponding plot of UPDRS3 in the outcomes panel shows only the 109 matching patients, represented by the lines with steep upward slopes. The mean UPDRS3 at Y3 of the group of selected patients (right arrow) is greater than that of the total patient sample (left arrow). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
3.4 Users can filter the 109 previously selected patients (in Figure 3.3) by gender by clicking the corresponding bar in the demographics panel. After selecting female patients (left) with unusually rapid changes in cognitive function as measured by SDM (center), users can then change the color scheme to explore the year of birth of the selected patients (right). . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
3.5 After applying the gender filter, the plots show that female patients (top) have comparable motor symptoms (UPDRS3) and general PD severity (T-UPDRS) at BL and less severe symptoms at Y3 than male patients (bottom). For semantic fluency (SFT), female patients show less severe symptoms both at baseline and Y3. Arrows mark the mean values at Y3 for the males and females. . . . . . . . . . . 66
3.6 (a) A filter that selects patients with increasingly greater baseline RBDQ. (b) As the filter moves to select patients with more severe baseline RBDQ (top 50%, 25%, and 10% from left to right), the mean T-UPDRS at Y3 increases from 50.17 to 53.65 and 58.54. . . . 68

4.1 Three nonlinear activation functions: sigmoid (blue), hyperbolic tangent (orange), and rectify (green). . . . . . . . . . . . . . . . . . . . 75

4.2 The techniques used in MRBrainS segmentation challenge shift from Random Forest (blue) and SVM (green) to deep-learning-based techniques (orange) from 2013 to 2016. Lower scores indicate higher performance. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77

5.1 The running time (m) of our hierarchical segmentation algorithm is considerably lower than F&H [3] and FSEG [4]. Both image size ($x$-axis) and time ($y$-axis) are in log scale. For large images, F&H and FSEG terminated prematurely because of insufficient memory. . . . . 81

5.2 An overview of the segment tree building procedure. Noise-resistant local binary patterns and superpixels are generated for each tile at all pyramid levels. A hierarchical graph-based segmentation subdivides each parent segment into child segments. . . . . . . . . . . . . . . . . 82

5.3 This example shows the calculation of $LBP_{8,1}$ using eight sample values, $g_1$ to $g_8$, taken on a unit circle (left). With the values shown in the right hand side, the corresponding LBP is a binary string `10001111`, which is 143 in decimal (right). . . . . . . . . . . . . . . . 84

5.4 The two-dimensional projections (d–f) show the distribution of sample blocks of (a) intestine, (b) cartilage, and (c) muscle tissues. The projections are obtained by MDS analysis based on the pairwise distances computed using (d) LBP, (e) NRLBP, and (f) joint distribution of intensity and NRLBP. . . . . . . . . . . . . . . . . . . . . . . . . 89

5.5 (a) The dissimilarity of two adjacent superpixels (represented by edge weight) is calculated as the chi-square distance of two corresponding joint histograms. (b) The edge between the blue and red segments is eligible to merge the two segments if its weight $w_6$ is less than or equal to $\min(\max(w_1, w_2, w_3) + \tau(4), \max(w_4, w_5) + \tau(3))$. . . . . . . . 91

5.6 The 11 types of tissues in the zebrafish embryo (P4). Image courtesy of Faas et al. [5]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94

5.7 With our visualization tool, clicking the yellow segment in the first row subdivides the top-right region of the zebrafish embryo into segment 5 (blue) and segment 6 (dark slate gray). Another click on segment 5 subdivides segment 5 into segment 26 (brown) and segment 32 (red). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96

5.8 The texture differences in the down-sampled grayscale images are more distinguishable with color highlights. First column shows the boundary between the muscle, intestine, and pronephric duct tissues; second column shows the boundary between muscle tissues. . . . . . . 97

5.9 First column shows the separation of region with spikes in the leg from abdomen; second column shows the segmentation of protopod, pleomeres, and pleopods. . . . . . . . . . . . . . . . . . . . . . . . . . 98

6.1 This example shows (a) an out of focus image and (b) a in focus image taken at the same time ($t = 1$) but at a different depth. The out of focus image provides much less useful information because the features are blurry. . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
6.2 Our CNN has three groups of layers. Each group includes two convolutional layers (red), each with 64 kernels, followed by a max-pooling layer (blue) that reduces the spatial resolution (including depth) by half. For simplicity the drawing only shows the $x$ and $y$ spatial dimensions, omitting the depth dimension. The outputs of the first and second fully-connected layers (green) are 200 and 2 dimensional; they correspond to the high-level pixel representations for visualization and the probabilities of cell states. . . . . . . . . . . . . . . . . . . . . . . 109
6.3 After extracting 200 deep features, we assign each pixel $p$ a color $(R_p, G_p, B_p, A_p)$ based on the user-defined transfer function, which can be easily modified after the 200 features are spectrally ordered along the $x$-axis. . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
6.4 The change in color (from green to red) of cell 39 from frame 49 to frame 54 suggests that the cell is dying. . . . . . . . . . . . . . . . . . . 112
6.5 This example shows the $200 \times 200$ feature-to-feature similarity matrix, in which bright entries indicate similar feature pairs, created using (a) arbitrary order and (b) spectral order. Similar features are closer to each other after organizing features in the spectral order, thus resulting in large bright regions in the similarity matrix. . . . . . . . 115
6.6 (a) The state change of a cell is visualized as a segmented line. The change in color from green to red indicates a state change of cell 39 from live to dead between frame 49 and 54. (b) The change in color of cell 39 can also be visualized separately with three lines, representing the RGB values over time. Here we can see the state change happens inside of the orange box. . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.7 This example shows the color annotations in four time points (a) 12, (b) 49, (c) 54, and (d) 78. Most cells are alive (green) at time point 12. They die (red) gradually in subsequent time points (i.e. 49, 54, and 78). Cell 39 (white arrow), which changes color from green to red, is dead sometime during frame 49 and 54. . . . . . . . . . . . . . 119
6.8 Our visualization allows users to select and compare cell trajectories without consulting different frames. This example shows three cells with distinct trajectories. Cell 3 stays alive whereas cells 38 and 39 died at specific frames, hinted by the drop in color green (and the growth in color red). . . . . . . . . . . . . . . . . . . . . . . . . . . 120

7.1 (a) Conventional workflow requires users to adjust both the feature space and the configuration. (b) Our deep-learning-assisted approach derives from labeled examples a feasible feature space automatically, effectively removing the need for an user-defined feature space. . . . . 128

7.2 (a) In the hippocampus dataset, we find it challenging to differentiate mitochondrial regions (orange) from non-mitochondrial regions using conventional intensity-based feature spaces. (b) The ground truth mitochondria. (c) The isosurface of isovalue 112, which is the average intensity value of mitochondria, does not correspond well to the boundary of mitochondria. In fact, no single isovalue would be suitable for differentiating the mitochondria (c.f. Figure 7.3a). (d) Our visualization result is comparable to the ground truth because it is based on a feasible feature space derived automatically. . . . . . . . 130

7.3 (a) The overlap in the intensity histogram shows that we cannot extract the mitochondria precisely using a single isovalue. The density plots, in which higher saturation indicates higher density, show that (b) intensity and gradient, and (c) seven selected Haralick features do not help extract the mitochondria either. The seven Haralick features are projected onto the first two principal components, "PC 1" and "PC 2", obtained using principal component analysis. . . . . . . . . . 132

7.4 Our proposed deep-learning-assisted approach first extracts high-level features from a trained CNN. We then use vector quantization to encode the extracted high-dimensional features of each voxel by the nearest centroid found using $k$-means clustering. Users modify the visualization result, which is generated by a marching cubes-based rendering, either by editing the characteristic feature vector of a target subvolume or exploring a pre-generated subvolume hierarchy semi-automatically. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134

7.5 Because the high dimensionality of feature space, we generate a binary scalar field by comparing the dot product of the characteristic feature vector $\mathbf{u}$ and the feature vector $\mathbf{v}$ of each voxel with a threshold $t$. After that we apply the conventional marching cubes algorithm on the binarized volume. . . . . . . . . . . . . . . . . . . . . . . . . . . 138

7.6 (a) Before spectral ordering, the similarity matrix, in which a bright pixel represents a pair of highly correlated features, does not show apparent pattern. After spectral ordering, highly correlated features are closer to each other. (b) A simple characteristic feature vector. (c) The arbitrary order of features does not exploit the correlation among features, thus leading to broken surfaces. The spectrally ordered features allow creating a visualization that reveals interesting structures using the same characteristic feature vector. . . . . . . . . 142

7.7 The size of codebook $k$ in VQ controls the amount of information loss. The zoom-in views shown in (b–e) are generated from the same subvolume (red box in (a)) in the bacteria dataset with $k$ ranges from 64 to 512. When $k = 64$, the view shows a noticeable disparity near the center of the spore (black box in (b)). The results in (c–e) are comparable without significant visual differences. . . . . . . . . . . 144

7.8 (left) The bacteria dataset is divided into the left and right halves for training and testing. (right) Both the spores (green) and the vesicles (yellow arrow) in the two sporulating bacteria are low in intensity. Because the difference in intensity between a spore and other structures (e.g. vesicle and cytoplasm) can be small, conventional intensity-based feature spaces will not differentiate them well. . . . . 149
7.9 This example shows the spore (green) in (b) the ground truth labels and the segmentation results (c) generated without postprocessing, and postprocessed with (d) class prior and (e) fitted transform for the raw image in (a). Simply scaling the predicted probability by class prior resulted in many false negatives for the spore class. In contrast, the data-driven fitted transformation of probability is more robust and generates a result comparable to the ground truth. . . . . . . . . 151
7.10 (a) The visualization of the spores in the bacteria dataset generated using the deep-learning-assisted approach. We show in the next two figures (b) the spore and (c) the cell wall of a bacterial cell. By rendering both semi-transparent surfaces at a time, (d) the composite result shows both the spore and the cell wall. (e) The characteristic feature vectors used to generate the green ($t = 0.36$) and orange ($t = 0.46$) surfaces. . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.11 (left) The hippocampus dataset consists of two volumes for training and testing (not shown here), both of the same size. (right) Both the mitochondria (orange) and membranes (yellow arrow) are low in intensity, thus making them inseparable using intensity-based features. 153
7.12 (a) Within the non-mitochondria region (outside of the orange regions), subregions with noticeably different characteristics can be identified. (b) The extracted deep features enable the detection of the non-mitochondria regions without membranes (blue subvolumes in the left). (c) The two characteristic feature vectors correspond to the mitochondria (orange, t = 63), and the non-mitochondria regions without membranes (blue, t = 49) in the composite visualization. . . 155
7.13 A slice in the volume for the (a) T1, (b) T1c, (c) T2, and (d) Flair channels. (e) The ground truth labels of the necrosis (red), edema (green), non-enhancing tumor (blue), and enhancing tumor (yellow). . 157
7.14 (a) The visualization of ground truth labels. (b) The composite result generated using the semi-automatic method. (c) The binary tree allows hierarchical exploration of the volume. The four child subvolumes (represented by the leaf nodes) are created by partitioning the parent subvolume (represented by the purple node). . . . . . . . . . . 159
7.15 (a) The reconstruction error of vector quantization decreases as the size of codebook (i.e. number of clusters) increases. (b) Larger codebook requires much more time running $k$-means and VQ. . . . . . . . 162

8.1 The CNN contains the encoding part and the decoding part. The numbers enclosed in the parentheses represent the numbers of output feature maps. A "/2" or "x2" indicates a change in spatial resolution after the (blue) block. See Figure 8.2 for details of each type of blocks. 170
8.2 The (a) input block, (b) encoding block, and (c) decoding block used in our CNN. We use batch normalization (BN) [6], Parametric Rectified Linear Unit (PReLU) [7], and dropout [8] in the three building blocks. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171
8.3 We replace the subsampling (top-right) by the stochastic downsampling (bottom-right) when the encoding block reduces spatial resolution. This stochastic operation introduces augmentations at the feature level inside of a residual block. . . . . . . . . . . . . . . . . . . 174
8.4 We can control the degree of augmentation by varying the value of $t$. Here we compare the result of (a) conventional subsampling and (b–d) stochastic downsampling with increasing value of $t$. The mean error (ME) comparing (a) and (b–d), caused by the stochasticity, increases from (b) 7.11, (c) 8.23, to (d) 11.33. Pixel values are in the scale of $[0, 255]$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175
8.5 This example compares the segmentation results of U-Net (orange), 2D CNN (model D, green), and 3D CNN (model H, blue), with the ground truth (purple). The 3D CNN showed promising result in which the errors are mostly near the ambiguous object boundaries. . 180

# Chapter 1: Introduction

In this dissertation, we present visual analytics tools and computational techniques for various types of complex biomedical data, including protein reaction networks, longitudinal multidimensional clinical data, and microscopy images. For each type of data, we present visualization strategies and techniques to address the challenges arising from unique data characteristics and properties. As data rapidly grow in size, there is an increasing need for visual analytics methods that combine the complementary strengths of computational techniques and human vision for identifying patterns and anomalies in large datasets. By keeping users in the cycle of knowledge discovery, the analysis is driven both by the data and by the knowledge, hypotheses, and intuitions of domain experts. Such a close collaboration between a computer and a human will lead to a better understanding of data and generate relevant insights.

## 1.1 Visualization of Rule-based Reaction Networks

In Chapter 2, we present *Simmune NetworkViewer* for exploring, validating, and debugging rule-based protein reaction models. These models contain rules that define the dynamics of protein interactions; the triggering of one rule can trigger a

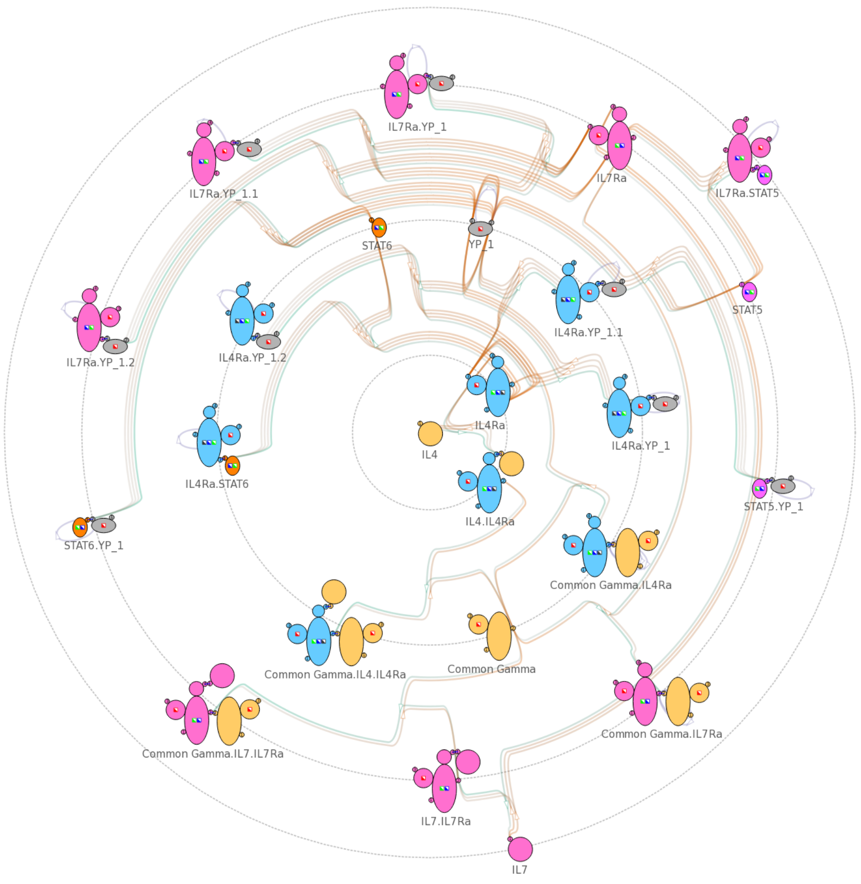


Figure 1.1: The circular layout of a network put `IL4` at the center and other nodes from the center to the peripheral with respect to their distances to `IL4`.

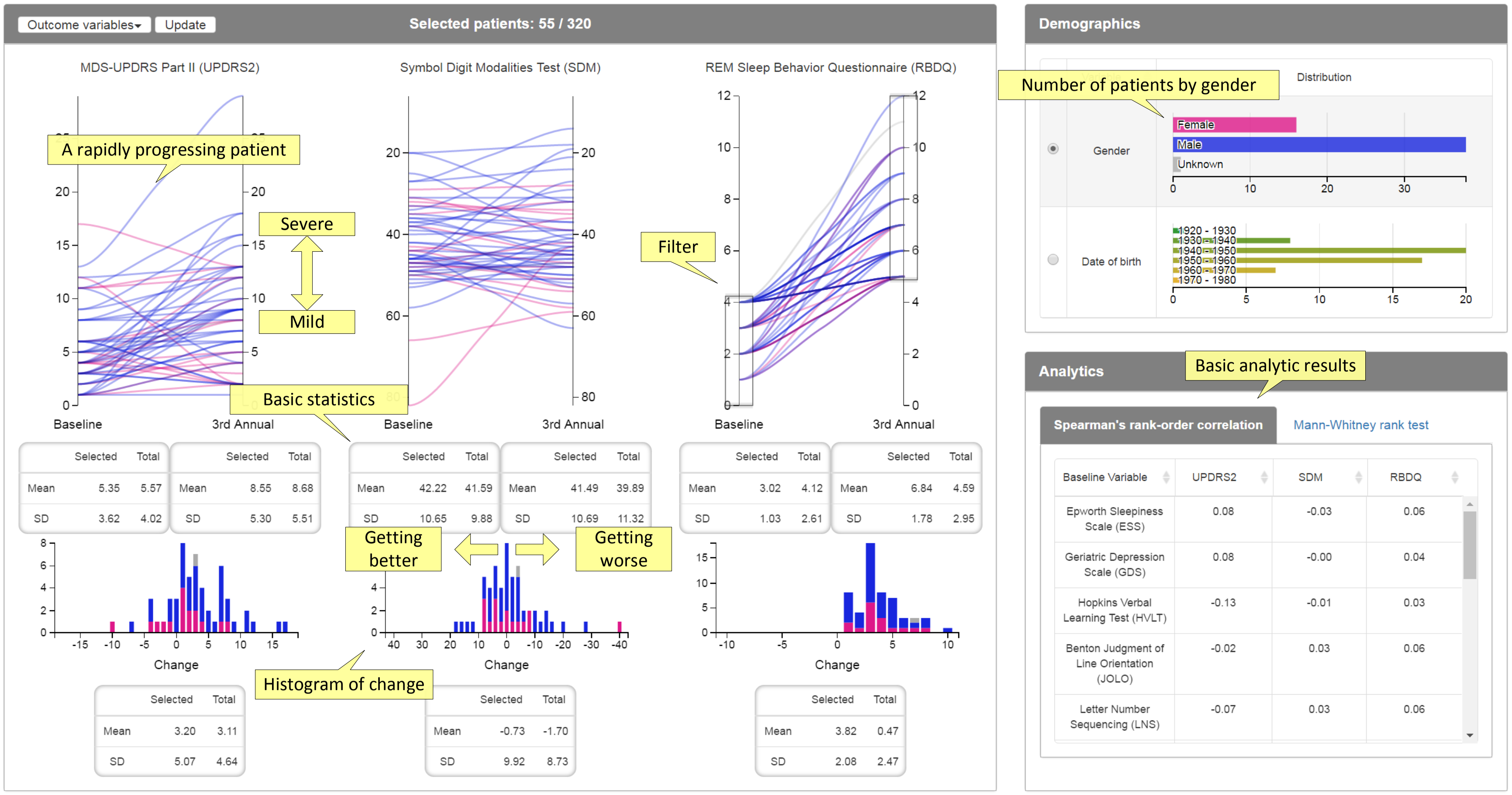
chain of other rules. For example, after binding with a ligand, a receptor can raise its energy level and subsequently trigger a series of intracellular signals. Visualizing network models is challenging because of the numerous potential interactions among various proteins. Furthermore, these interactions are better interpreted when presented alongside the global context of the model. Although several existing visualization tools targeted rule-based models, they require users to read textual rule specifications to extract relevant rule-specific information. This requirement limits the efficiency and efficacy of these tools, especially for users who are not expert modelers.

Instead of using textual representations, our approach for rule representations is a visual and symbolic one (Figure 1.1). This way, users can extract relevant information intuitively. Based on this visual representation, we create concise visualizations showing the inter-relationships of various interacting components. In our network visualization, we show only a simplified network overview after omitting rule-specific details, and then present those details only upon request; this detail-on-demand strategy presents information selectively and directly on top of the network view, thus enabling users to comprehend and compare rules and their pre- and post-conditions in the global context of network. We have shown that our tool improves the comprehensibility of complex network models. For example, Figure 1.1 shows that, in the cytokine signaling model, the nodes interacting with `IL4` are near the center whereas those interacting with `IL7` are in the peripheral; the positions of nodes reveal the implemented behaviors in the model.

## 1.2 Visualization of Temporal Changes in High-dimensional Clinical Data

In Chapter 3, we present *Winnow* for visualizing longitudinal multidimensional Parkinson's disease data. In this dataset, each patient at a time point is represented as a high-dimensional feature vector, which corresponds to the severity of various symptoms. A straightforward way of analyzing such data is applying clustering and dimension reduction techniques. The unique challenge in analyzing clinical data, however, is the gap between computational techniques and clinicians, who are usually unfamiliar with those techniques. In our preliminary study, we found that conventional computational techniques do not always generate interpretable or clinically meaningful results. This finding is consistent with past studies that showed the instability of existing pure data-driven Parkinson's disease analyses. Without engaging clinicians more in the cycle of knowledge discovery, this gap will continue to impede the exploration of potential patterns that lead to actionable clinical results.

In this study, we collaborated with Dr. Lisa Shulman, Dr. Ann Gruber-Baldini, Dr. Rainer von Coelln, and other clinicians and neurologists to close this gap. We design an interactive tool, Winnow, that facilitates clinicians to investigate data with minimal training and technical background. Winnow generates simple and intuitive visual representations to engage clinicians more in the analysis. By representing patients as lines connecting two endpoints, each corresponds to a time point, users can easily inspect disease progression visually by looking at the slope of lines (top left in Figure 1.2). The lines are color-coded to facilitate the comparison of patients in different demographic groups (e.g. female and male). Winnow supports highly flexible and interactive filtering of patients based on multiple variables. For example, REM sleep behavior disorder (RBD) is an important preclinical marker for early diagnosis of Parkinson's disease. The muscles of patients diagnosed with RBD remain active during sleep; this abnormal behavior causes patients to act out their dreams. In Figure 1.2, we select 55 patients (out of 320) with low baseline RBD questionnaire (RBDQ) and high RBDQ at the third annual visit. We evaluate Winnow with example case studies analyzing a public Parkinson's disease dataset and show promising results.



UPDRS2 Baseline

| | Selected | Total |
|---|---|---|
| Mean | 5.35 | 5.57 |
| SD | 3.62 | 4.02 |

UPDRS2 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 8.55 | 8.68 |
| SD | 5.30 | 5.51 |

UPDRS2 Change

| | Selected | Total |
|---|---|---|
| Mean | 3.20 | 3.11 |
| SD | 5.07 | 4.64 |

SDM Baseline

| | Selected | Total |
|---|---|---|
| Mean | 42.22 | 41.59 |
| SD | 10.65 | 9.88 |

SDM 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 41.49 | 39.89 |
| SD | 10.69 | 11.32 |

SDM Change

| | Selected | Total |
|---|---|---|
| Mean | -0.73 | -1.70 |
| SD | 9.92 | 8.73 |

RBDQ Baseline

| | Selected | Total |
|---|---|---|
| Mean | 3.02 | 4.12 |
| SD | 1.03 | 2.61 |

RBDQ 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 6.84 | 4.59 |
| SD | 1.78 | 2.95 |

RBDQ Change

| | Selected | Total |
|---|---|---|
| Mean | 3.82 | 0.47 |
| SD | 2.08 | 2.47 |

| Baseline Variable | UPDRS2 | SDM | RBDQ |
|---|---|---|---|
| Epworth Sleepiness Scale (ESS) | 0.08 | -0.03 | 0.06 |
| Geriatric Depression Scale (GDS) | 0.08 | -0.00 | 0.04 |
| Hopkins Verbal Learning Test (HVLT) | -0.13 | -0.01 | 0.03 |
| Benton Judgment of Line Orientation (JOLO) | -0.02 | 0.03 | 0.06 |
| Letter Number Sequencing (LNS) | -0.07 | 0.03 | 0.06 |

Figure 1.2: Our visual analytics tool, Winnow, consists of three panels: the Outcomes panel (left), the Demographics panel (top right), and the analytics panel (bottom right). The colors represent the gender of patients (magenta for females and blue for males).

## 1.3 Biomedical Image Understanding

In the rest of this dissertation (Chapters 4–8), we focus on biomedical image analysis, which is especially challenging because biomedical images have low signal-to-noise ratio and contrast. In Chapter 4, we briefly review these challenges and introduce convolutional neural networks (CNNs), which we use extensively as classifiers and feature extractors in Chapters 6–8.

### 1.3.1 Visualization of Microstructures in Gigapixel Microscopy Images

In Chapter 5, we address a common problem when exploring gigapixel images: the disparity between screen and image resolution. In practice, users would first inspect an image at low resolution and then zoom-in to specific regions of interest for high-resolution details. Such an exploration method is a compromise at best because zooming in and out routinely is tedious and inefficient, especially for large images. Furthermore, this routine is ineffective for texture analysis because relevant texture information is lost after downsampling–users would not notice regions with subtle texture differences when viewing at low resolution. Because many targets in biomedical image analysis are distinguishable only by their microstructures and textures, we must address the resolution disparity problem when inspecting gigapixel biomedical images visually.

We present a visualization that highlights texture differences across various

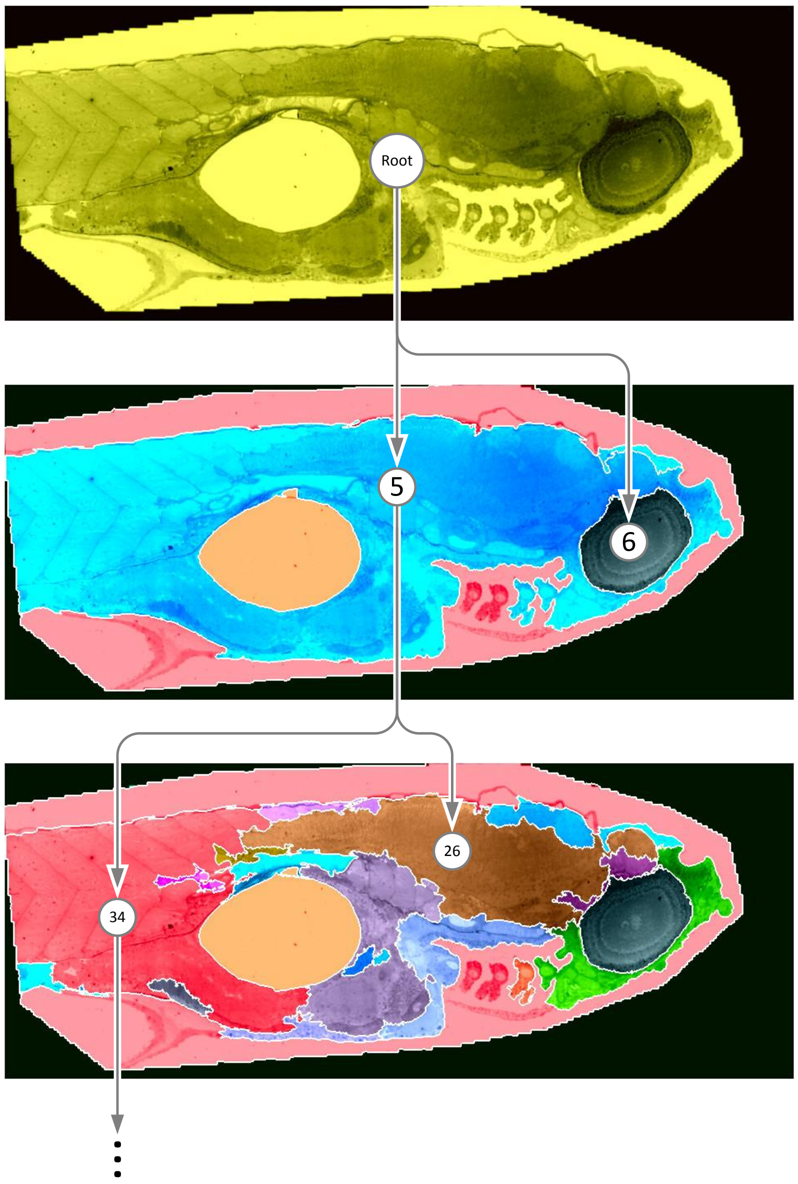


Figure 1.3: With our visualization tool, users can change the color highlights by traversing the segment hierarchy; adjacent segments are assigned distinguishable colors such that they are discernible at coarse resolutions.

scales based on a highly efficient hierarchical segmentation method. The segmentation method at each scale partitions the adjacency graph of superpixels into segments, thus reducing the memory footprint. We assess the similarity of a pair of adjacent superpixels based on the joint distribution of intensity and noise-resistant local binary patterns; we empirically show that this combination of features improves the separation of different microstructures in electron microscopy images. We assign different colors to adjacent segments such that texture differences can be seen even when viewing at low resolutions. Using our visualization tool, users can interactively and hierarchically inspect alternative segmentations at different scales. For example, in Figure 1.3, clicking the yellow segment in the first row subdivides the top-right region of the zebrafish embryo into segment 5 (blue) and segment 6 (dark slate gray). This exploration strategy helps users locate regions of interest more efficiently.

### 1.3.2 Visualization of Live-cell Imagery

In Chapter 6, we present visualization techniques for time-lapse live-cell imagery. As a common practice in live-cell imagery, multiple images are taken at various depths at each time point to accommodate for vertical cell movements and focus drift problem. For a specific time point, however, only a few (in-focus) images are useful; the rest of the images provide little additional value and can be safely discarded. Visually inspecting all images (of various depths) at a time point is difficult because of the inflated data size. In addition, inspecting all frames from the

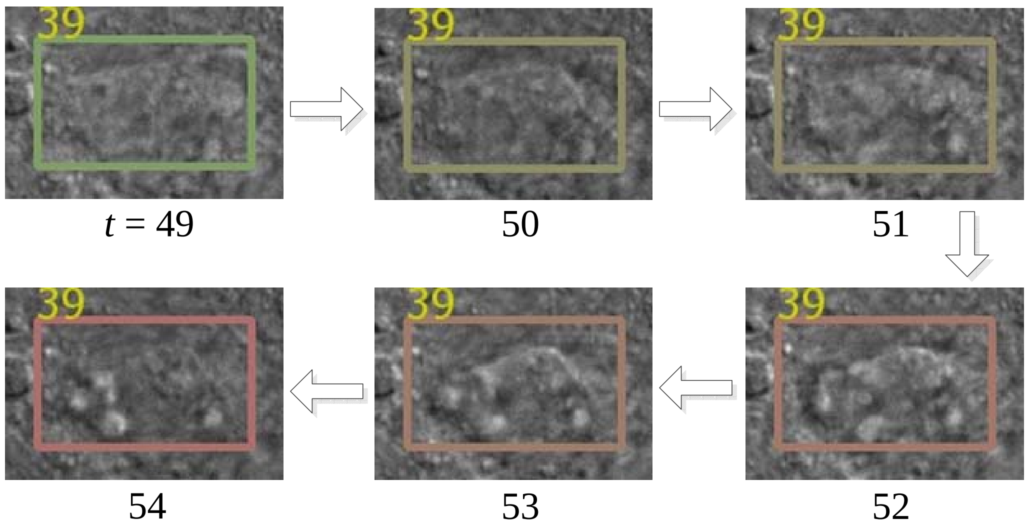


Figure 1.4: The change in color (from green to red) of cell 39 from frame 49 to frame 54 suggests that the cell is dying.

beginning to the end leads to a huge cognitive load. For example, users must track cell movements and detect cell states (e.g. alive or dead) simultaneously.

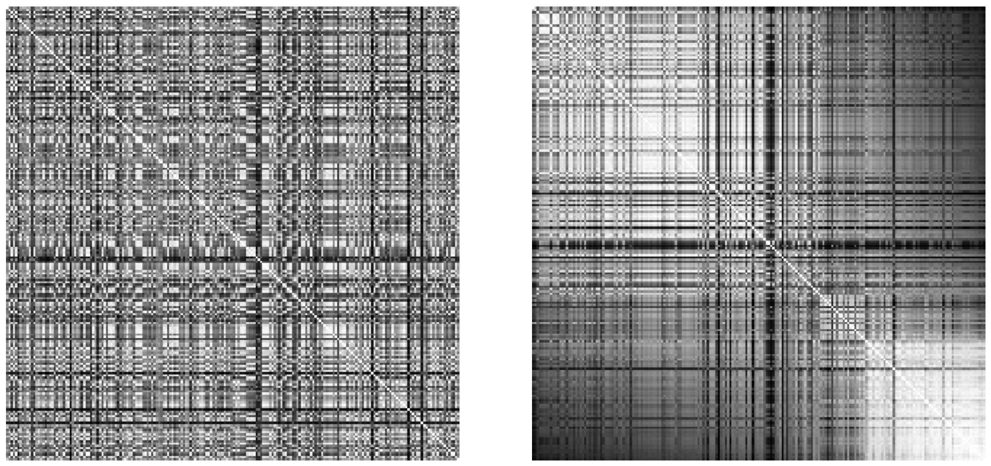
We use 3D convolutions in a CNN to learn features across different depths for detecting cell states. Because of the redundancy at each time point, the CNN must learn to extract and combine only useful information. After training, we use the CNN as a feature extractor that generates deep features corresponding to high-level concepts related to cell states. Based on these deep features, we build visualizations that depict changes in cell states, which are difficult to describe using hand-crafted and conventional local features. Although the CNN-derived features are abstract, we show that users can still design useful visualizations when the extracted features are organized based on pairwise similarities. Our visualization represents cell states using colors assigned by a user-mediated method that converts deep features into colors; the generated color annotations facilitate the identification of cells that went from live to dead. For example, the change in color annotations in Figure 1.4

indicates that the cell died between time points 49 and 52.

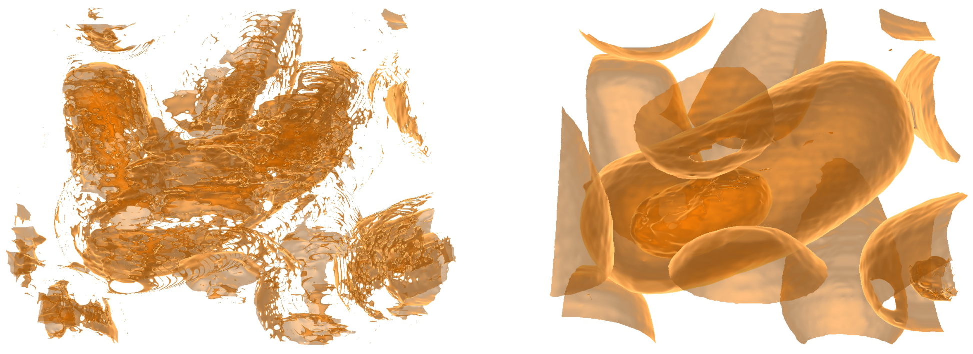
### 1.3.3 Deep-learning-assisted Volume Visualization

In Chapter 7, we present a similar CNN-based technique for volume visualization. Conventional techniques rely on handcrafted, local features to distinguish voxels that belong to different objects and structures. For example, intensity and gradient magnitude are used for separating two objects of different intensity values. Nevertheless, many complex structures in biomedical volumes can be separated only by high-level features. This lack of separation challenges the creation of informative visualizations with which users can inspect different structures visually. Furthermore, configuring the volume visualization requires significant computations and user interactions. As a result, interactive volume visualization remains a challenging problem for large, complex volumetric datasets.

To alleviate the need for handcrafted, local features, we extract deep features derived from a trained CNN. Based on this CNN-based voxel representation, we use the marching-cubes algorithm to extract surfaces of complex objects after classifying voxels based on a user-defined configuration. We apply the same technique that organizes features into an accessible order to facilitate the definition of such configurations. For example, Figure 1.5 shows that the surface of the cytoplasm of a cell can be depicted easily when the extracted features are organized by their similarity. Alternatively, we present a hierarchical exploration method that groups similar voxels using spectral clustering. This way, we further simplify user interactions



(a) Similarity matrix (Left: arbitrary order. Right: spectrally ordered)



(b) Visualization (Left: arbitrary order. Right: spectrally ordered)

Figure 1.5: (a) Before spectral ordering, the similarity matrix, in which a bright pixel represents a pair of highly correlated features, does not show apparent pattern. After spectral ordering, highly correlated features are closer to each other. (b) The arbitrary order of features does not exploit the correlation among features, thus leading to broken surfaces. The spectrally ordered features allow creating a visualization that reveals the cytoplasm of the cell.

to expanding and collapsing trees that correspond to splitting and merging groups of voxels. We implement our volume rendering method on a GPU, thus allowing changing the configurations at interactive speed. We show that our visualization technique is successful in depicting complex objects (e.g. cytoplasm in Figure 1.5) in multiple datasets that are challenging for conventional methods.

### 1.3.4 CNN-based Segmentation of Volumetric Microscopy Images

Finally, in Chapter 8, we present techniques to improve CNN-based segmentation of volumetric images when given scarce training samples. Although CNNs have been successful in natural image applications, part of the success is attributed to large natural image datasets, which enable the training of large and deep CNNs with numerous trainable parameters. Applying CNN techniques to biomedical images is particularly challenging because of the limited size of training data; the scarcity is, however, inevitable because both collecting biomedical images and labeling target structures are time-consuming and expensive. Conventional data augmentation techniques expand the training set by generating similar samples from existing ones. Nevertheless, standard operations such as random rotation and scaling may generate samples that are inappropriate for biomedical applications.

Instead of data augmentations, we present a stochastic downsampling technique to perform augmentations at the feature level. This augmentation technique adds spatial distortions directly to the features inside of a residual block and can be used together with data-level augmentations. In Table 1.1, we show empirically

Table 1.1: Segmentation results obtained by U-Net and eight network variants (A–H). The number of parameters is in millions.

| Model | #Parameters | Precision | Recall | Jaccard |
|---|---|---|---|---|
| U-Net [1] | 36.97M | 99.3 / 94.8 | **99.7** / 87.6 | 99.0 / 83.5 |
| A (2D, Full, –) | 1.68M | 99.5 / 91.4 | 99.5 / 91.6 | 99.1 / 84.4 |
| B (2D, Full, A) | 1.68M | 99.5 / 93.0 | 99.6 / 90.8 | 99.1 / 85.0 |
| C (2D, Fact., –) | 0.60M | 99.6 / 91.9 | 99.5 / 93.1 | 99.2 / 86.1 |
| D (2D, Fact., A) | 0.60M | 99.6 / 92.5 | 99.6 / 93.1 | 99.2 / 86.5 |
| E (3D, Full, –) | 4.94M | **99.7** / 92.5 | 99.6 / **93.9** | 99.3 / 87.2 |
| F (3D, Full, A) | 4.94M | 99.6 / 93.2 | 99.6 / 93.4 | 99.3 / 87.5 |
| G (3D, Fact., –) | 0.63M | **99.7** / 92.7 | 99.6 / **93.9** | 99.3 / 87.4 |
| H (3D, Fact., A) | 0.63M | 99.6 / **95.0** | **99.7** / 93.3 | **99.4** / **88.9** |

[*] The numbers separated by slashes correspond to the results for non-mitochondria (left) and mitochondria (right)
[**] The best results are marked in bold.

that the feature-level distortions introduced by stochasticity can improve segmentation performance significantly. The results also favor factorized convolutions, which reduce the number of parameters significantly by using only low-rank kernels, thus improving model generalization and prevent overfitting. In addition, our segmentation results suggest that, although using more parameters, 3D convolutions that combine information along all three spatial dimensions (including the $z$-axis) are superior to 2D convolutions (model E–H v.s. model A–D in Tabel 1.1, see also Section 8.7). For the tested dataset, the 3D CNNs always outperform the 2D counterparts in both precision and recall regardless of the different settings. Figure 1.6 shows an example where 2D CNNs misclassify a non-mitochondria region in the top left corner as mitochondria.

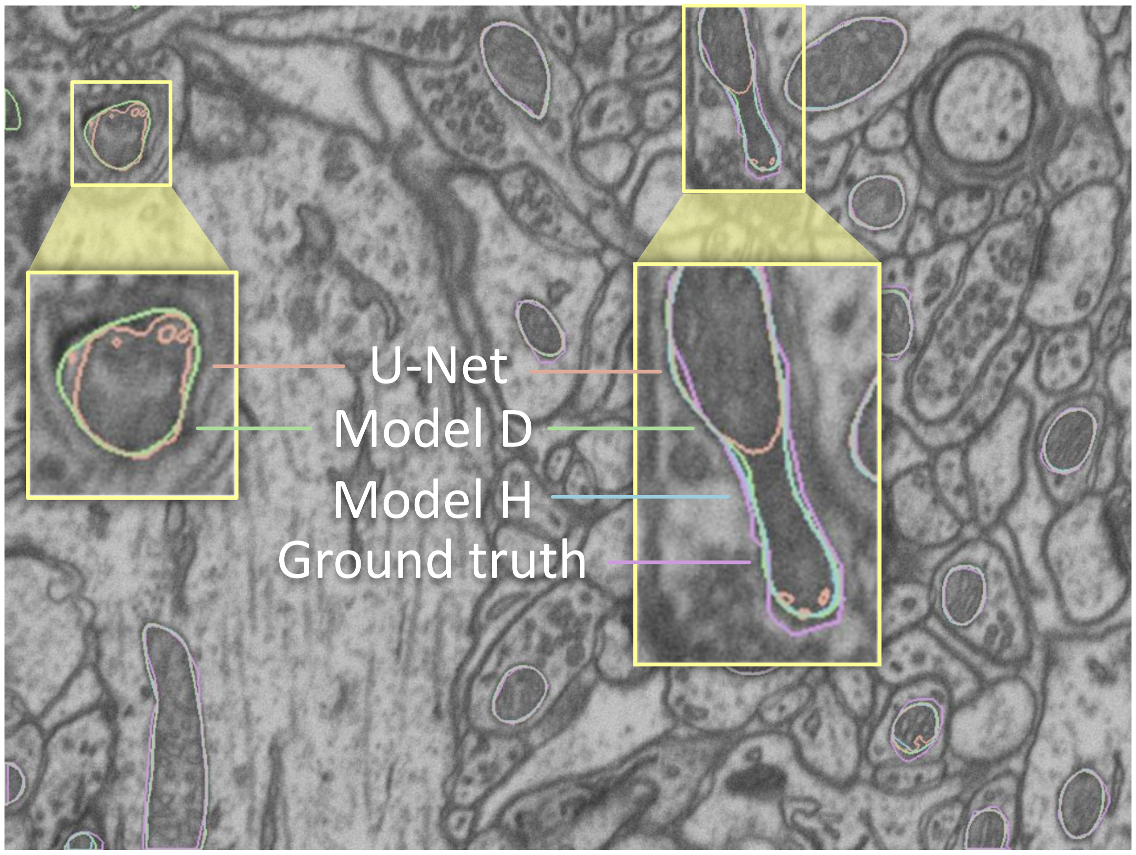


Figure 1.6: The segmentation generated by model H, which uses 3D convolutions, is comparable to the ground truth; the segmentations generated by model D and U-Net [1], both use 2D convolutions only, misclassify the non-mitochondria region near the top left corner as mitochondria.

# Chapter 2: Visualization of Rule-based Reaction Networks

## 2.1 Introduction

Network representations of complex cell-biological signaling processes contain numerous interacting molecular and multi-molecular components that can exist in, and switch between, multiple biochemical and/or structural states. Showing the interaction categories (i.e. associations, dissociations, and transformations) in such networks is nontrivial because their specifications involve information such as reaction rates and conditions with regard to the states of the interacting components. The additional complexity in network presentation leads to the challenge of having to reconcile competing visualization objectives: providing a high-level overview without omitting relevant information, and showing interaction specifics without overwhelming users with too much detail displayed simultaneously.

Existing tools typically address this challenge by splitting the information into several categories that are rendered separately through combinations of visualizations and/or textual and tabular elements; this strategy requires network modelers to consult several sources to obtain comprehensive insights into the underlying assumptions of the model. As model complexity grows, the cognitive load of analyzing the inter-relationships among interacting components also grows significantly. By taking advantage of the visual language of the *Simmune Modeler*, an integration of these two aspects (i.e. overview and interaction specifics) into a single display can help relieve the hustle. The *Simmune* package [9, 10, 11] is a framework of computer programs that allows researchers to build, simulate, and analyze quantitative models of cellular signaling processes. Because users can create models in Simmune using only iconographic symbols, the software is easily accessible to both theorists and non-theorists. In contrast to other approaches, the *native* representation of the reaction rules here is thus a visual, symbolic one. This visual representation allows a highly efficient method for rendering protein reaction networks, addressing the preeminent challenge for network visualization, namely combining high-level overviews with details provided on-demand.

In the following, we present our visual analytic tool, *Simmune NetworkViewer*, for the visualization of protein reaction networks. This tool first creates a general network view showing all user-defined molecular complexes and the features determining their possible states (e.g. the potential to carry phosphorylations or to assume specific conformations). These complex *prototypes*, or, in the language of Simmune, *complex species*, that do not carry any particular states are linked by the biochemical network resulting from the structural interaction possibilities among their molecular binding sites. Within this view, users can select reactions to access details such as the particular states the participating complexes are in when the reactions occur and what their resulting states are; those details provide complete specifications of reaction rules embedded directly into the network view without context-switching. Users can also search for reaction rules that meet specific criteria, such as belonging to a certain reaction category or including certain types of reacting species. Importantly, the tool presents search results as an overlay on top of the network view, thereby showing the reactions within their biochemical contexts.

## 2.2 Related Work

Software tools such as Cytoscape [12], Osprey [13], and VisANT [14] are widely used to analyze genetic networks and pathways. These tools provide a variety of filtering methods and visualizations for visual inspection. Typically, the networks being analyzed consist of nodes (e.g. genes) that are connected by lines if they represent entities that show correlated behavior.

Other methods have been developed specifically for visualizing cell biological protein reaction networks where the nodes can contain additional inner structure and the links between them indicate biochemical processes. For example, nodes and links in a network can represent multi-molecular complexes and their reactions. The Systems Biology Graphical Notation (SBGN) [15] project, for example, provides a well-documented standard for visualizing biological processes, including protein interactions. It offers three different views that visualize aspects such as the flow of information (activity flow), entity relationship diagrams and can provide diagrams giving information about the sequence of biochemical modifications components in the network undergo.

Molecular Interaction Maps (MIMs) [16, 17] aim at combining as much information as possible in a single diagram. However, a comprehensive visualization of all reactions, including involved binding sites, molecular states, and the precondition and postcondition for the reactions is possible only for rather small networks. For large networks, users have to simultaneously trace multiple lines to infer complete reaction specifications; this requirement can render the process of parsing complex interaction diagrams cumbersome.

Much of the complexity of reaction networks arises from the fact that molecules and pairwise molecular interactions frequently participate as elements in several multi-molecular complexes. Reducing model definitions back to this fundamental level, rule-based modeling approaches offer concise ways to specify molecular interactions, their conditions and consequences [18, 10, 19]. Several iconographic representations of such rules have been suggested [20, 21, 22].

Using the rule-based BioNetGen language (BNGL) [18], the visualization tool RuleBender [23] addresses the conflict between readability and completeness by linking a contact map depicting possible interactions between molecular binding sites with BNGL code elements of the full rule set from which the contact map is derived. Their approach represents a significant step forward but comes at the cost that the visualization itself contains only part of the information. Interactions and states have to be selected to access additional information via the textual mode of BNGL. Users are thus required to learn the model description language, which may impede communication between modeling experts and experimental biologists not familiar with BNGL. Another rule-based approach, Extended Contact Maps [24], provides more detailed information but also follows the strategy of omitting reaction aspects in favor of increased readability. The additional information that is necessary to understand a particular reaction can be retrieved from accompanying textual explanations of the labels in the map.

The rxncon software [25] takes a modular approach to visualizing reaction networks at various levels of complexity by separating *elemental reactions* - that take molecular complexes as input and modify them through reactions - from *contingencies* that specify under which conditions these reactions may occur. Based on various ways of combining the information in these two categories, the rxncon software can generate several different pathway visualizations, including SBGN based graphs, with varying degree of completeness with regard to rendering the assumptions of the underlying models. This modular approach results in highly efficient visualizations of various aspects of interaction networks. Nevertheless, users have to consult *reaction graphs* or *reaction lists* together with *contingency lists* to fully access the conditions for and the consequences of reactions.

The approaches discussed so far have in common that their network visualizations either become very complicated as models grow or (for the rule-oriented approaches) that they separate the display of molecular reactions from the information regarding the conditions under which those reactions occur.

## 2.3 Simmune Framework

Our network visualization is built upon the iconographic symbols used in the Simmune framework. The Simmune framework consists of several modeling tools including a tool for specifying molecular properties and interactions (the Simmune

Modeler), a cell morphology design application, and a simulator. Using the graphical interface of the Simmune Modeler [22], researchers define molecules, their components (sub-domains) and binding sites as well as interactions between such binding sites and how they depend on the states of the interacting molecules. In the following, we briefly introduce the visual language and the terminologies in the Simmune Modeler using a simple ligand-receptor reaction model as an example. In this example, a receptor is embedded into the cytoplasmic membrane and consists of an extracellular and an intracellular domain. When the extracellular domain binds to its ligand, the intracellular domain switches its state from inactive to active, allowing it to interact with other molecules inside of the cell, thereby initiating an intracellular signaling process.

### 2.3.1 Visual Representation of Reaction Rules

The Simmune Modeler uses unique symbols to represent molecules, molecules components, and binding sites. To represent different molecule component states and binding site statuses Simmune uses the icons listed in Table 2.1. Molecule components can be assigned several squares representing state variables that can be "on", "off" or "don't care" and may represent, for example, phosphorylations or conformational states, depending on the nature of the molecules and their interactions. Circles represent binding sites (with site indexes displayed inside the circles). A filled circle represents a bound site, possibly with a blue line connecting the other bound site when specified. Future releases of the Simmune Modeler and

Table 2.1: The icons used for different molecule component states and binding site statuses.

| Molecule component state | Icon |
|---|---|
| Undefined (complex species only) | ◩ |
| On | 🟥 |
| Off | □ |
| Don't care | ⬚ |
| **Binding site status** | **Icon** |
| Undefined (complex species only) | ① |
| Bound | ❶ ❷–❶ |
| Unbound | ① |
| Don't care | ⓵ |

the NetworkViewer will support alternative symbolic representations using icon libraries based on existing visualization approaches such as the one used in the STKE database of signaling pathways (http://stke.sciencemag.org/cm/), or the SBGN style [26].

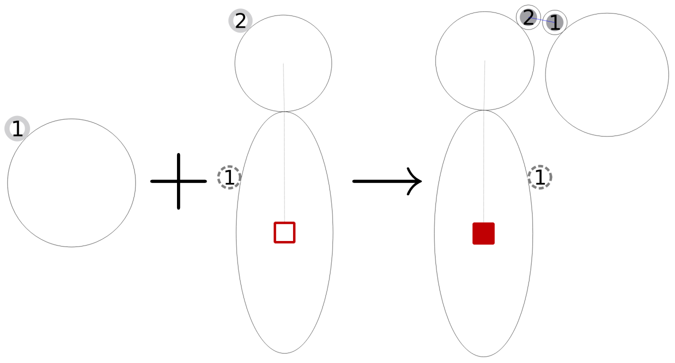
A *complex species* comprises a specific set of structurally identical *complexes* that are constructed with the same set of molecules and binding site interactions. Within a species, the complexes differ only with regard to the states of their components. We can therefore consider a complex species to be a prototype describing a particular set of complexes that are structurally identical, whereas a complex is an "instance" of the complex species it belongs to. This hierarchy of structural and state-specific information about molecular complexes is critical for the Simmune NetworkViewer to generate concise reaction network visualizations. In the rest of this chapter, "species" and "complex species" are used interchangeably.

Simmune builds reaction networks automatically from the specification of bimolecular reaction rules. Depending on their characteristics, reaction rules belong into one of the three categories: complex association, complex dissociation, and complex transformation. Although reaction rates are crucial to the specification, we omit them here for simplicity.

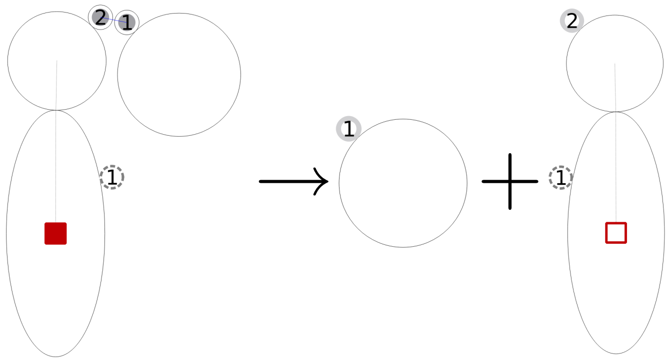
For example, receptor ligation is an association reaction where a ligand binds a receptor, inducing a change in the receptor's conformational and functional state. Figure 2.1a shows a complex association where the `Ligand` binds the extracellular domain of the `Rec inactive` complex and produces the `Ligated Receptor` complex. The receptor's intracellular molecule component state changes from "off" to "on", reflecting the change in the receptor's state from inactive to active. For consistency, we refer to "complexes" even if they consist of single molecules when defining reactions.

Ligand dissociation is a reaction that dissociates the ligand from the receptor by removing the bond between them. Figure 2.1b shows a complex dissociation where the reacting complex `Ligated Receptor` breaks into two product complexes, the `Ligand` and the `Rec inactive` complex, after the bond between the receptor and the ligand is dissolved. The receptor's molecule component state changes from "on" to "off" to reflect its deactivation.

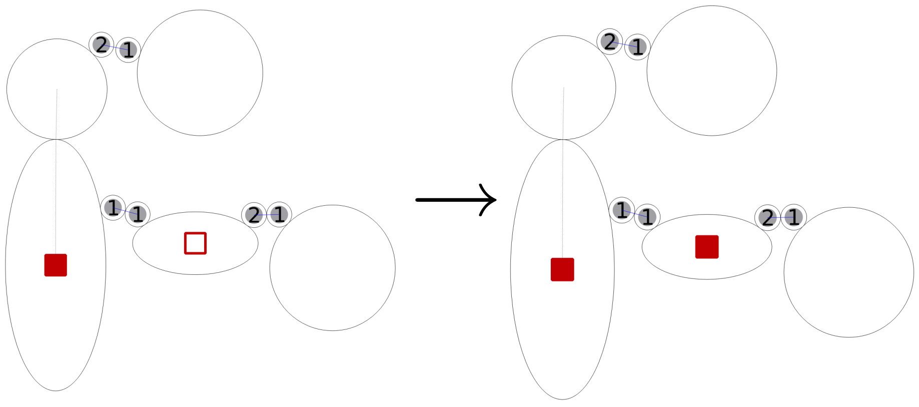
To include an example of molecule transformation reaction, we allow the intracellular domain of the activated (ligand-bound) receptor to interact with a G-protein and enzymatically catalyze the replacement of Guanine Diphosphate (GDP) at the G-proteins' G$\alpha$ subunit through Guanine Triphosphate (GTP). Figure 2.1c shows the visual representation of this complex transformation mediated by the receptor that changes the G$\alpha$ state from GDP to GTP. This is reflected by the switch of the



(a) Complex association



(b) Complex dissociation



(c) Complex transformation

Figure 2.1: (a) A complex association where the two complexes on the left-hand side, `Ligand` and `Rec inactive`, bind and produce a `Ligated Receptor` complex. (b) A complex dissociation where `Ligated Receptor` on the left-hand side splits into a `Ligand` and a `Rec inactive` complex. (c) A complex transformation where the reacting complex `LigReg_Gab_GDP` transforms into the product complex `LigRec_Gabg_GTP`.

"GTP" state (i.e. represented by the square in the horizontal ellipse depicting G$\alpha$) from "off" to "on".

## 2.4 Visual Analytics Tool: Simmune NetworkViewer

### 2.4.1 Network Graph

The Simmune NetworkViewer generates and visualizes a network graph, which is a directed bipartite graph composed of two categories of nodes: complex species nodes and intermediate nodes, the latter representing reactions. The total number of nodes in the graph thus equals the number of complex species plus the number of reactions.

In the network graph, there exists an edge between an intermediate node and a species node if and only if the corresponding reaction involves, as reactant or product, a complex of the corresponding species. If the involved complex is a reactant (e.g. in the left-hand side of the reaction), the edge goes from the species node to the intermediate node. If that complex is a product (e.g. in the right-hand side of the reaction scheme), then the edge goes from the intermediate node to the species node.

The example G-protein model encompasses eight complex species and eleven reactions, including those mentioned previously in Figure 2.1. The corresponding Figure 2.2 shows a network graph with 19 nodes and 29 edges. We will describe the layout and visual design in detail later.

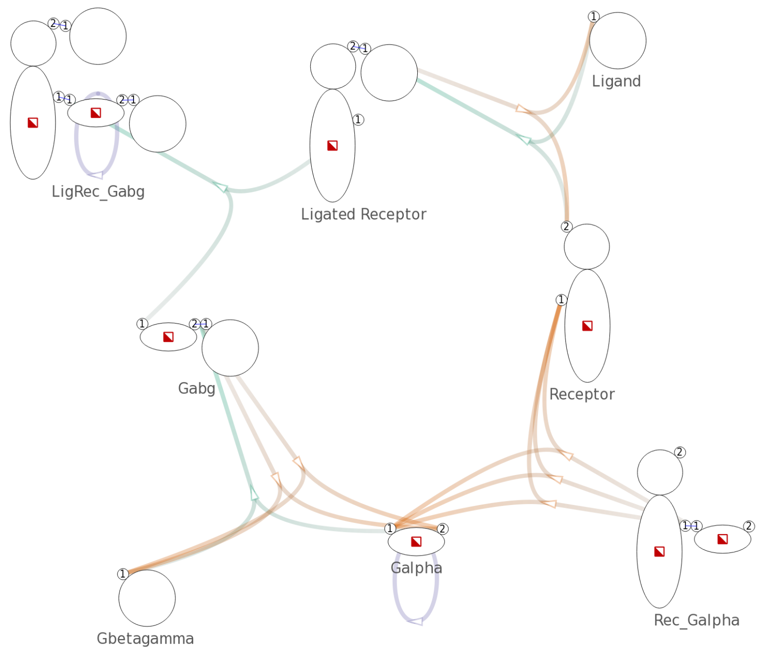


Figure 2.2: Overview of a G-protein network with 19 nodes and 29 edges created with a model that consists of 8 complex species and 11 complex reactions. Complex species nodes are displayed with the iconographic representation used in the Simmune framework. Intermediate nodes are displayed as small arrows indicating direction of reactions.

To optimize the efficiency of displaying network information the viewer uses two main layout principles:

1. Create a node for each complex species instead of each complex with specific biochemical properties.

   Creating nodes for all biochemically (as opposed to structurally) distinct complexes and linking them through arrows indicating reactions would frequently generate an overwhelming number of nodes in the network graph with severely limited readability and strong node overlap. Because complexes of the same species are merely different in the molecule component states and binding site statuses, we can present only the complex species within the network overview, and provide complex- and reaction-specific information upon user request.

2. Introduce intermediate nodes to represent reactions.

   In principle, reactions could be indicated as edges between complex species nodes. Doing so would, however, result in confusing edges when there are multiple reaction rules between a pair of complex species. This is quite a common situation because pairs of molecular complexes may have multiple interaction possibilities that are modulated by their biochemical properties.

### 2.4.2 Visual Representation of Network Graph

#### Node representation

We display complex species nodes using the iconographic representation used in the Simmune modeling framework, thereby providing a concise and consistent visualization. The name of a species is shown under the corresponding species node. We use small arrows to represent intermediate nodes functioning as reaction handles. The arrows also serve as indications of the direction of reactions. See Figure 2.2 for an example.

#### Edge representation and layout

We use different hues to distinguish types of reactions and variation in color saturation (i.e. from less saturated to more saturated) to indicate the direction of edges. As a default, we use green for complex associations, orange for complex dissociations, and purple for complex transformations. See Figure 2.2 for an example.

However, users can freely specify colors for different types of reactions.

A highlighted edge has greater opacity and width. The tool tip on an edge shows the reaction rate of the corresponding reaction.

Each edge is rendered as a Bézier curve. Edges that represent complex associations or dissociations have one of their endpoints pointing to the binding sites involved in related reactions. Note that complex transformations do not involve any binding sites, therefore related edges point to the center of species nodes.

For example, in Figure 2.2, the species node `Receptor` has two binding sites. Five edges, representing five reactions, connect the species node `Receptor`: three edges point to the first binding site and two edges point to the second.

### Network layout

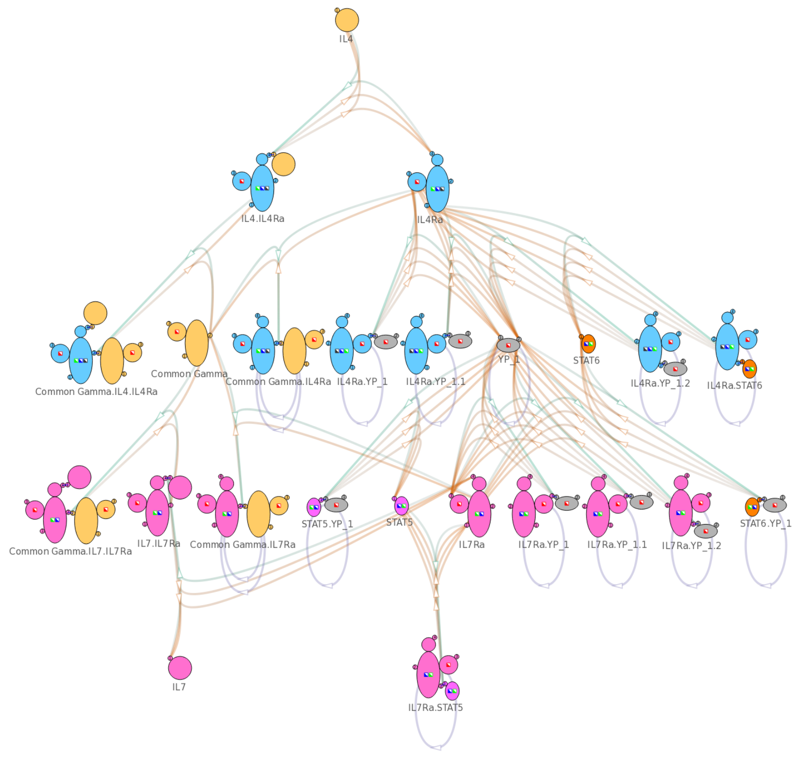
The NetworkViewer provides three network layout types: non-hierarchical layout, level-based layout, and circular layout. Whereas the non-hierarchical layout provides a general overview of networks, exploiting the hierarchy in networks and reorganize network layout accordingly is useful in creating meaningful visualizations. Similar to the orderly MIMs proposed in [27], we construct level-based and circular layout based on the hierarchy generated after defining a reference point in the network. Users may switch among different layouts depending on the analysis they wish to perform.

### Non-hierarchical layout

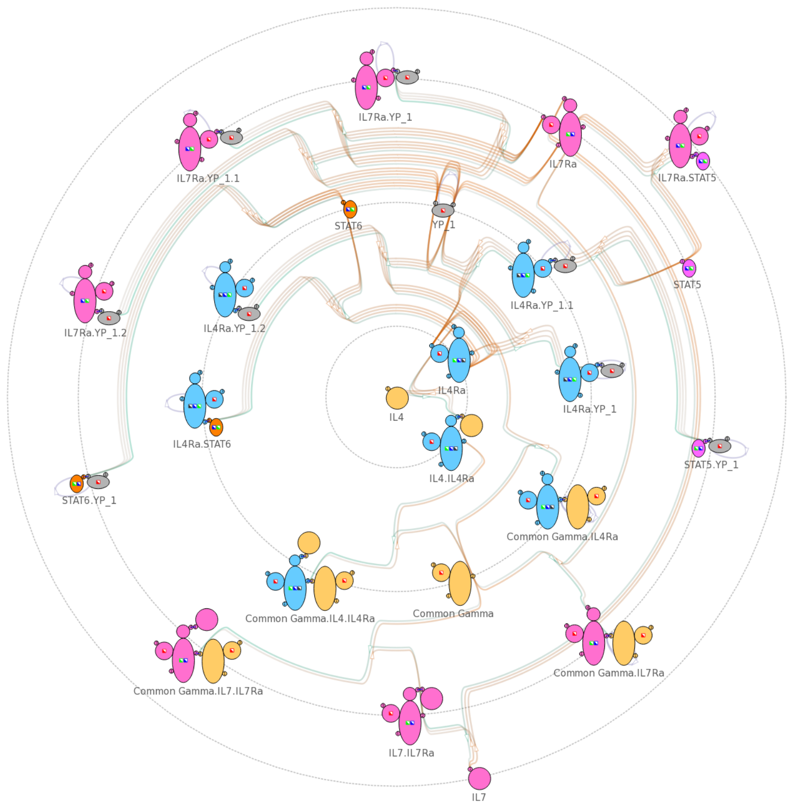
We use the NEATO [28] layout algorithm of Graphviz [29] to generate a positional layout for the nodes in the network. After experimenting with different overlap removal techniques available in Graphviz, we choose to eliminate overlaps by incorporating overlap removal constraints into the layout algorithm. A non-hierarchical layout of the network graph created from the G-protein model is shown in Figure 2.2.

### Level-based layout

In the level-based layout, nodes are arranged into levels with respect to their distances to the user-selected reference complex species node. Nodes with smaller distances (defined as the minimal number of reactions that lead from a complex to the reference node) are positioned closer to the top of the layout. The level layout is generated with the help of the DOT [30] layout algorithm of Graphviz. Figure 2.3a shows the level layout of a cytokine signaling model, incorporating receptors and downstream effectors for IL4 and IL7, with a reference species node IL4. The two cytokines, IL4 and IL7, and their respective receptors can be easily differentiated by color. The result of the level-based layout automatically separates the interacting species by the type of cytokines – those interacting with IL4 on the top and those interacting with IL7 at the bottom.



(a) Level layout



(b) Circular layout

Figure 2.3: After specifying the reference species `IL4`, the network can be reorganized using level-based and circular layout. (a) The corresponding reference node `IL4` is placed at the top level; the other nodes are arranged with respect to their distances to the reference node. (b) The corresponding reference node `IL4` is placed at the center; the other nodes are arranged with respect to their distances to the reference node.

## Circular layout

In the circular layout, the reference complex species node is fixed at the center and the rest of the nodes are arranged on concentric circles around this center. Similar to the criteria used in the level-based layout, nodes with smaller distances are positioned closer to the center (i.e. on a concentric circle with a smaller radius). We calculate the position of nodes in the circular layout with a conversion from Cartesian to polar coordinate given the result of the level-based layout. See Figure 2.3b for an example of the circular layout of the cytokine model with cytokine IL4 and its interacting species closer to the center, and cytokine IL7 and its interacting species on the periphery.

Passing estimates of node sizes to Graphviz allows the layout algorithms to minimize node overlap. Users can resolve residual overlap manually by adjusting the positions of nodes. In the models we tested, we found that users can resolve overlap in a short time.

The NetworkViewer saves the manually-adjusted layout as well as other visual attributes such as edge width in an auxiliary file, which, when provided along with the model file, guides the NetworkViewer to generate identical visualization using the stored configuration. We note that the functionality to save the changes to the automatically generated visualization may also help to convey information (e.g. for emphasizing certain network sections) as part of remote collaborations.

## Tree view and reaction list

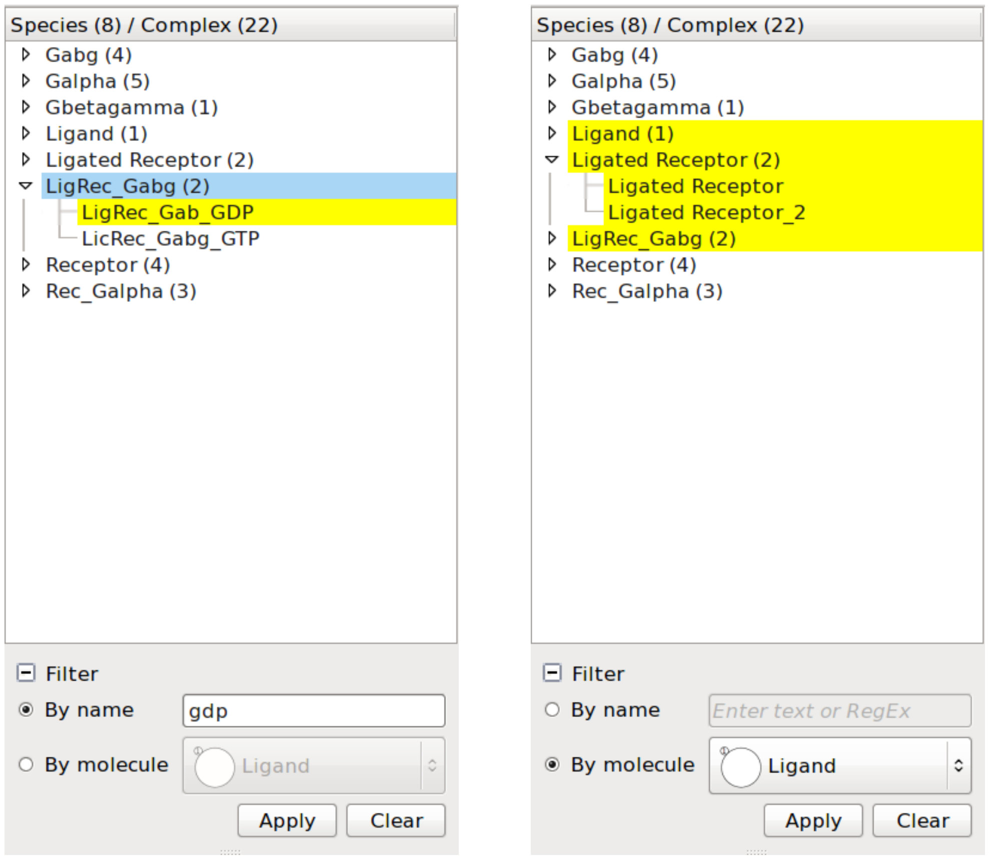
In addition to the aforementioned graphical network display, we show the species-complex hierarchy in a tree view. In another panel, we list all reactions grouped into the three reaction types (associations, dissociations, transformations). Selections performed in the tree view and reaction list are carried over into the graphical network display.

## Filtering

The NetworkViewer facilitates locating relevant complexes and/or complex species in the tree view by allowing users to filter by either (1.) complex name or (2.) component molecules.

1. The NetworkViewer highlights complexes and complex species whose names contain the specified term in yellow. When a complex species does not contain the term in its name but one of its child complexes does, the complex species is shown in light blue to indicate that it has at least one matching child complex that might be hidden in the collapsed list. See Figure 2.4a for an example of filtering by the term "gdp".

2. The NetworkViewer highlights complexes and complex species that contain the specified molecule in yellow. For example, Figure 2.4b shows that only three complex species: `Ligand`, `Ligated Receptor` and `LigRec_Gabg` contain the molecule `Ligand`. Note that a complex contains a molecule if and only if



(a): [✓ By name] gdp; By molecule Ligand

(b): By name; [✓ By molecule] Ligand

(a) Filter by name (b) Filter by molecule

Figure 2.4: (a) With the search term "gdp", the matching complex LigRec_Gab_GDP is highlighted in yellow. The non-matching complex species LigRec_Gabg has a matching complex and is therefore colored in light blue. (b) The complex species and complexes that contain the molecule Ligand are highlighted in yellow.

its species contains that molecule too.

### 2.4.3 User Interactions

After the initial automated layout process, the network graph (see the example shown in Figure 2.2 and Figure 2.3) provides an overview of the network model that offers an accessible abstraction at species level. Different types of specific information are presented upon user request.

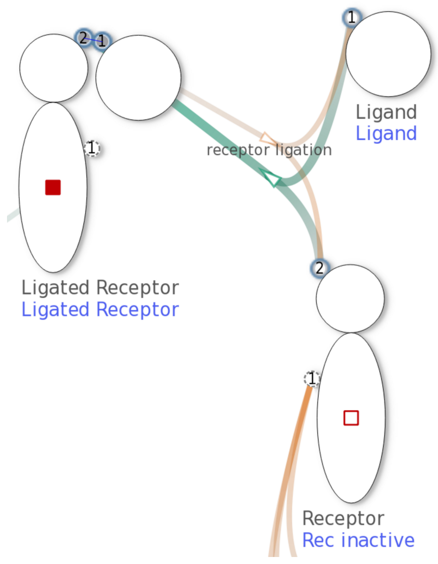
Within the layout, a complex species usually interacts only with complex species nearby. Users can zoom in and move to specific regions of interest. To focus on a complex species it can be selected by either clicking the complex species node in the network display or the corresponding item in the tree view. The selected complex species and the reactions in which it is involved are highlighted.

Reactions can be selected by clicking intermediate nodes (representing the reactions) in the network, or items in the reaction list. The NetworkViewer indicates selected reactions by highlighting all the related edges.

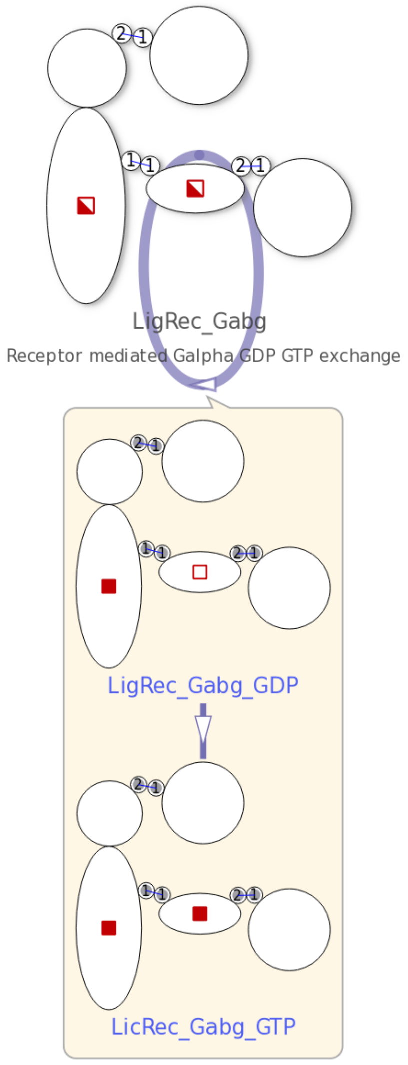
When the selected reaction is a complex association or a complex dissociation, the involved complex species nodes are depicted with their molecular states and binding site statuses according to the specified reaction rule. For example, the binding sites that prior to a selected association were unbound are now linked through bonds. The names of the complexes are added to the labels in blue beneath the name of the species. See Figure 2.5a for an example. If the selected reaction is a complex transformation, a hovering frame, as shown in Figure 2.5b, shows the initial and product complex.

A typical user query consists of identifying which reactions a particular complex is involved in. After the complex has been selected in the tree view it is highlighted in network visualization along with its reactions.

For example, in Figure 2.6, after selecting the complex `Rec inactive` the NetworkViewer highlights two reactions, which are shown in Figure 2.1a and Figure 2.1b, that involve `Rec inactive`.



(a) Select a complex association



(b) Select a complex transformation

Figure 2.5: (a) The three complex species nodes show the three involved complexes, `Ligand`, `Rec inactive` and `Ligated Receptor`, after the complex association `receptor ligation` (shown in Figure 2.1a) is selected. (b) A hovering frame shows the two involved complexes, `LigRec_Gab_GDP` and `LicRec_Gabg_GTP`, after the complex transformation `Receptor mediated Galpha GDP GTP exchange` (shown in Figure 2.1c) is selected.

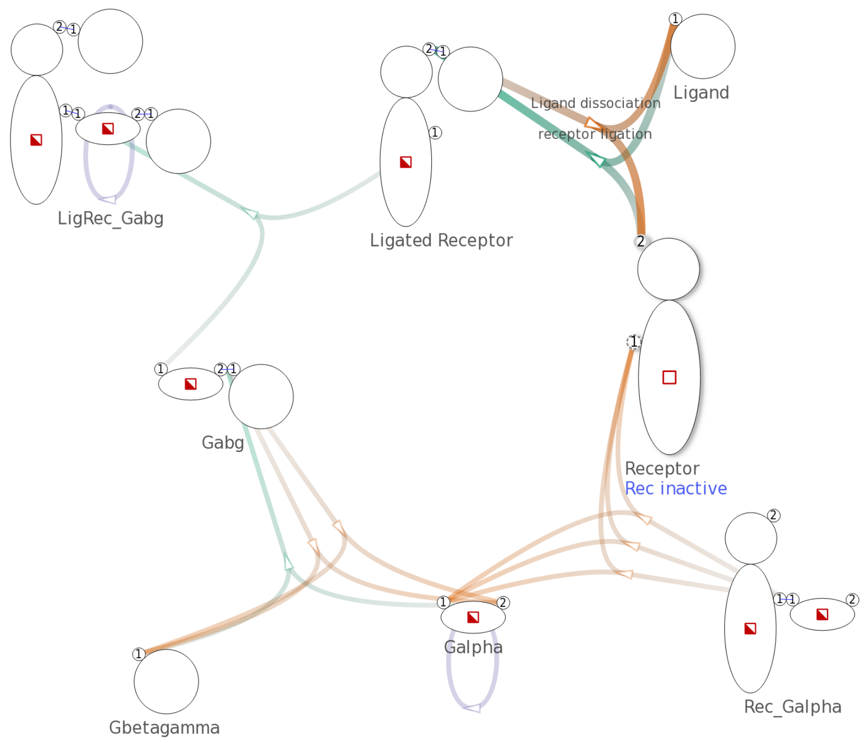


Figure 2.6: After selecting the complex `Rec inactive`, the related edges generated by the two reaction rules in Figure 2.1a and Figure 2.1b are highlighted. Because other edges in the network remain unchanged, the visualization shows that `Rec inactive` participates in the two aforementioned reactions only.

Searches can also be performed for complexes of a complex species that match a particular set of states. Such set of states could, for example, be combinations of phosphorylations on molecules carrying multiple phosphorylation states. The NetworkViewer finds and shows all reactions having a reactant or product complex that matches the constraint.

The complex species being searched is marked by a red border. Users can change the search constraint by clicking the squares that represent the states. The complexes that match the specified set of molecule component states will be selected. During the search both states “on” and “off” will match a user-defined query state “don’t care”. Figure 2.7 shows a search on the complex species `Receptor`. The

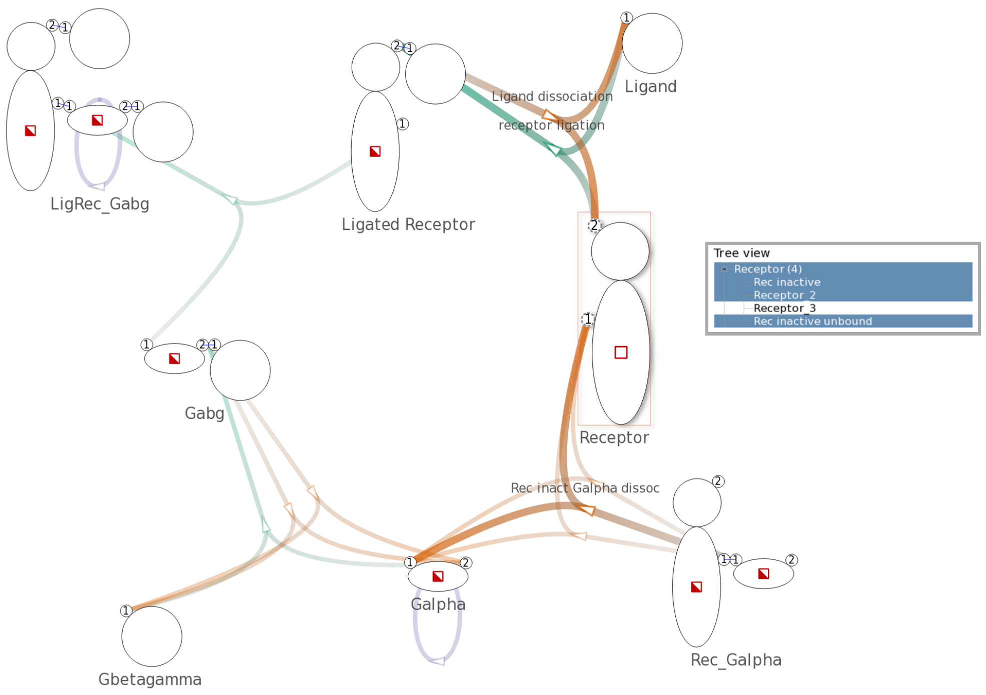


Figure 2.7: A search is performed on the complex species `Receptor` to find complexes with an "off" state in the intracellular molecule component. The edges are highlighted to show that three reactions contain a reactant or product complex with an "off" state in the intracellular molecule component.

specified search constraint is an "off" state in the intracellular molecule component. Three complexes, `Rec inactive`, `Receptor_2` and `Rec inactive unbound`, match the constraint. The matching complexes are involved in three reactions that are highlighted in the display.

### 2.4.4 Implementation

Simmune NetworkViewer is implemented in C++ and is released under a download agreement with the Simmune project[1] for academic use[2].

[1]http://www.niaid.nih.gov/labsandresources/labs/aboutlabs/lsb/Pages/simmuneproject.aspx

[2]The software may not be used for commercial purposes without prior permission from the NIAID Office of Technology Development. (Commercial license)

## 2.5 Case Study

To illustrate some of the capabilities of the NetworkViewer, we apply the tool to explore a model for the binding of the Epidermal Growth Factor Receptor (EGFR) to its binding partners. EGF provides proliferation, differentiation and survival signals and the membrane-bound EGF receptor is associated with several types of cancer if its expression or activation changes erroneously. The model we developed is based on the work by Hsieh et al. [2] addressing the possibility of multiple adaptors to bind to the same phosphorylated EGFR cytoplasmic (intracellular) domain simultaneously as opposed to competitively (or sequentially). Note that these constraints regarding the possible combinations of molecular interactions were obtained using coarse-grained modeling and may, thus, contain methodological artifacts. But our goal here is to illustrate the application of the NetworkViewer for visualizing networks based on interaction rules and the proposed constraints are very well suited to be implemented in a rule-based model. Following [2], an EGFR cytoplasmic domain in our model has four binding sites, 992, 1068, 1148 and 1173 that, when phosphorylated at the tyrosine residues, can mediate interactions with adaptor molecules Grb2, PLC$\gamma$1, Stat5 and Shc. For our model, we assume that the sites are, indeed, tyrosine-phosphorylated and assign the names pY992, pY1068, pY1148 and pY1173 to the sites, where the pY stands for Tyrosine-phosphorylated. Note that a more complete model of the EGF receptor would have to take into account that the receptor undergoes ligand-induced dimerization prior to activation (phosphorylation).

Stat5 and Grb2 can bind to site pY992 and pY1068, respectively. PLC$\gamma$1 can bind to pY992 or pY1173. Shc can bind to pY1148 or pY1173. These six interaction possibilities were translated into visually encoded reaction rules using the Simmune Modeler. In [2], the authors reported several binding constraints in this system. For example, once an adaptor PLC$\gamma$1 binds to pY992 or pY1173, it prevents another PLC$\gamma$1 from binding to the other, remaining, site. To accommodate these constraints in our model, we assigned two molecule component states "bndPLCg992" and "bndPLCg1173" to the `EGFR` species indicating whether a PLC$\gamma$1 is bound to either one of the two binding sites pY992 and pY1173, respectively. An additional state "bndSHC1148" is needed for the constraint that the binding of Shc to site pY1148 and the binding of PLC$\gamma$1 to site pY1173 are mutually exclusive. See Figure 2.8 for the visual representation of the complex species `EGFR`.

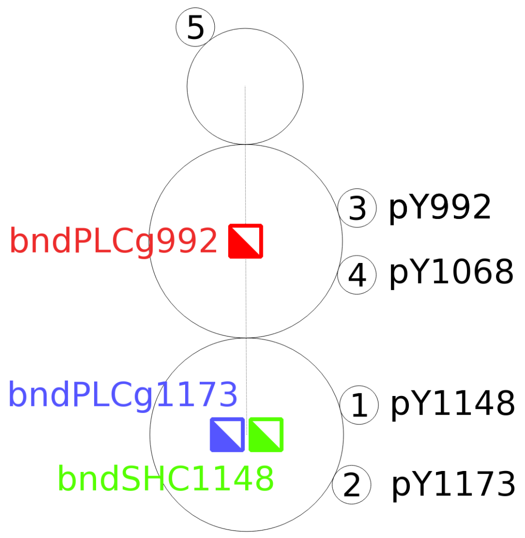
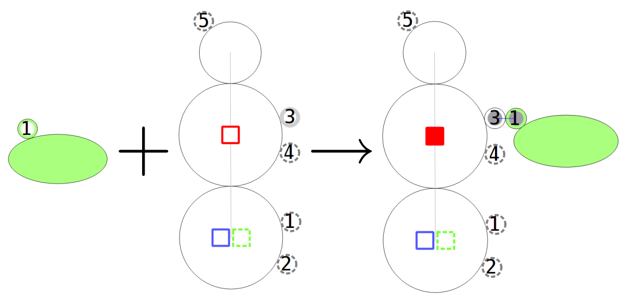
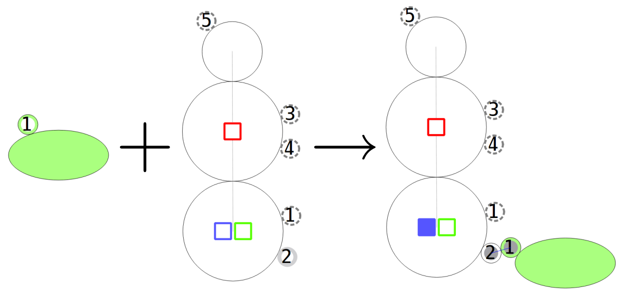


Figure 2.8: The complex species `EGFR` has five binding sites, four of them (e.g. with indices 1 – 4) can be used to bind adaptors. Three molecule component states, "bndPLCg992", "bndPLCg1173" and "bndSHC1148" accommodate the binding constraints reported in [2], which are described as rules defining which adaptors can bind simultaneously to the EGFR.



(a) Binding to site pY992



(b) Binding to site pY1173

Figure 2.9: (a) Two states “bndPLCg992” and “bndPLCg1173”, represented as red and blue squares, have to be “off” for PLC$\gamma$1 to be able to bind to site pY992. (b) All three states “bndPLCg992”, “bndPLCg1173” and “bndSHC1148”, represented as red, blue and green squares, have to be “off” for PLC$\gamma$1 to be able to bind to site pY1173.

The conditions for binding of PLC$\gamma$1 to the EGFR using the two possible sites are depicted in Figure 2.9. PLC$\gamma$1 can only bind to EGFR when both molecule component states “bndPLCg992” and “bndPLCg1173” are “off”. After ligation, the corresponding state – pY992 or pY1173, depending on which site PLC$\gamma$1 has bound to, switches to “on”, thereby blocking the other site for a second PLC$\gamma$1 molecule. As depicted in Figure 2.9b “bndSHC1148” must be in the “off” state to permit the binding of PLC$\gamma$1 to site pY1173.

After loading the model into the NetworkViewer, the network overview in Figure 2.10 shows the possible reactions between the adaptors and the EGFR as well as the binding sites these reactions involve. For example, PLC$\gamma$1 can bind

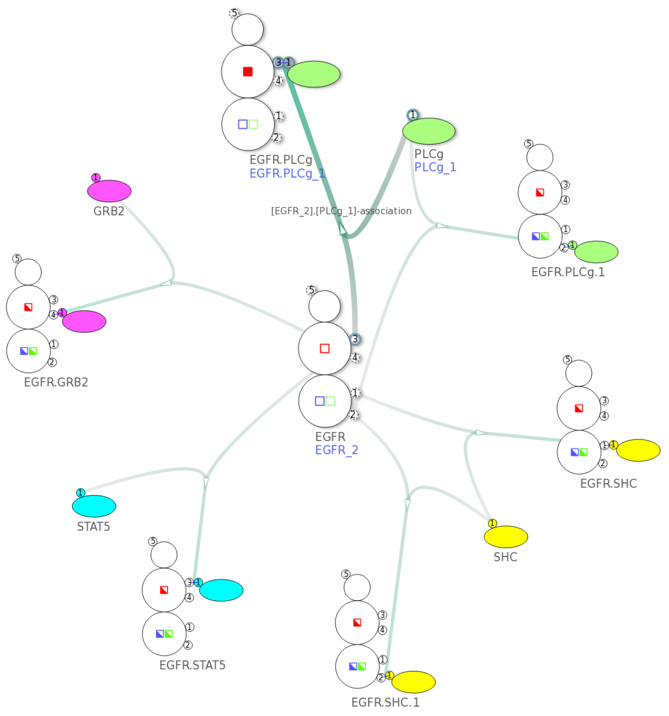
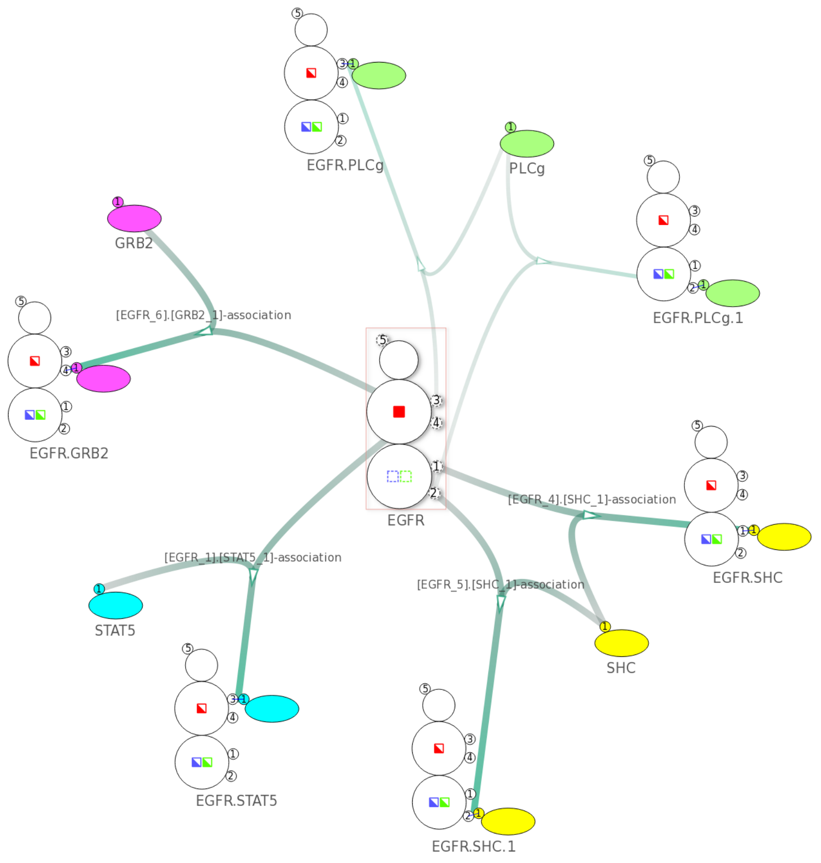


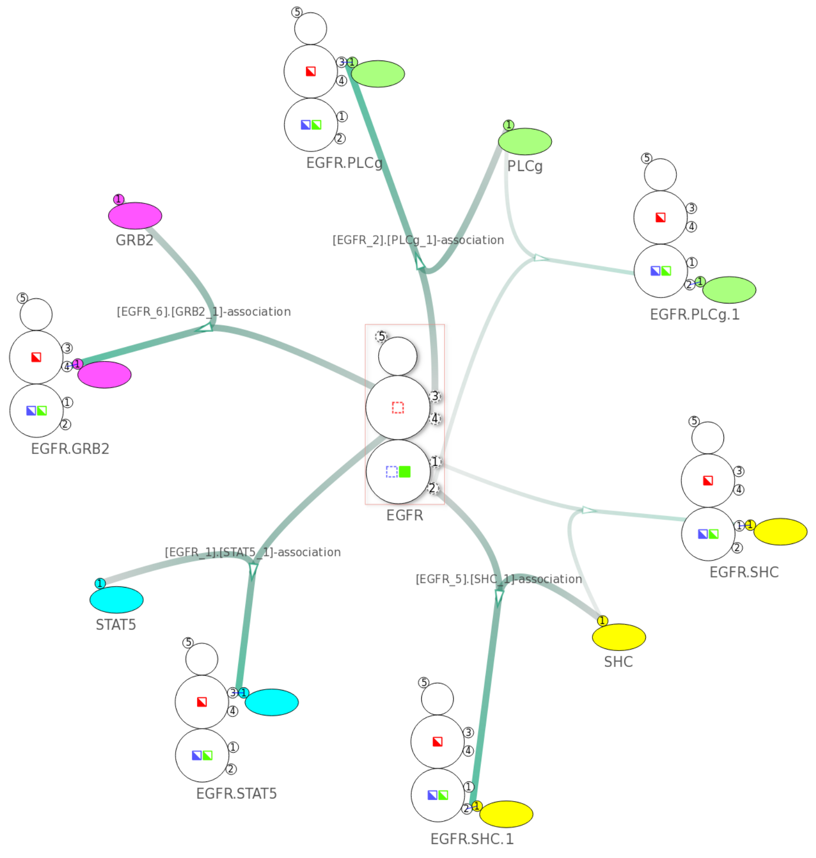
Figure 2.10: The network contains 17 nodes and 18 edges created from 11 complex species and 6 complex reactions. Here we select the complex association rule described in Figure 2.9a.

in two ways to the EGFR using two different binding sites. After selecting the corresponding intermediate node, the display shows that the binding of PLC$\gamma$1 to site pY992 changes the state "bndPLCg992" from "off" to "on".

We now verify the binding constraints in this model by searching for eligible rules given specific states of the EGFR. For example, whenever the state "bndPLCg992" is "on", no second PLC$\gamma$1 can bind to the EGFR (Figure 2.11a). Similarly, whenever the state "bndSHC1148" is "on" PLC$\gamma$1 cannot bind to site pY1173 (Figure 2.11b). Moreover, Shc cannot bind to site pY1148 either when the state "bndSHC1148" is "on". Because an "on" state of "bndSHC1148" indicates that Shc is already bound to site pY1148, there cannot be another Shc binding to the same site.


EGFR.PLCg
PLCg
GRB2
EGFR.PLCg.1
[EGFR_6].[GRB2_1]-association
EGFR.GRB2
EGFR
[EGFR_4].[SHC_1]-association
[EGFR_1].[STAT5_1]-association
[EGFR_5].[SHC_1]-association
EGFR.SHC
STAT5
SHC
EGFR.STAT5
EGFR.SHC.1

(a) Search for “bndPLCg992” is on


EGFR.PLCg
PLCg
GRB2
[EGFR_2].[PLCg_1]-association
EGFR.PLCg.1
[EGFR_6].[GRB2_1]-association
EGFR.GRB2
EGFR
[EGFR_1].[STAT5_1]-association
[EGFR_5].[SHC_1]-association
EGFR.SHC
STAT5
SHC
EGFR.STAT5
EGFR.SHC.1

(b) Search for “bndSHC1148” is on

Figure 2.11: (a) After specifying a search constraint where the state “bndPLCg992” is on, the visualization shows that PLC$\gamma$1 cannot bind to site pY992/pY1173. (b) After specifying a search constraint where the state “bndSHC1148” is on, the visualization shows that PLC$\gamma$1 cannot bind to site pY1173.

## 2.6 Conclusions and Future Work

In this chapter, we introduced the NetworkViewer as part of the Simmune modeling framework. The NetworkViewer provides an interactive network model visualization that facilitates efficient exploration of models built with the Simmune Modeler using the same visual language. Exploiting the hierarchical nature of the reaction network model, the NetworkViewer creates a compact model overview, in which only the complex species and complex reactions are displayed as nodes. User interaction activates the presentation of detailed information about, for instance, the molecule component states of a complex participating in a particular reaction. The case study of a simple model of interactions among the EGFR cytoplasmic domain and its binding partners illustrates how the network overview and user interaction options of the NetworkViewer can be used for an efficient navigation of model components and interaction conditions, here provided as adaptor binding constraints.

Our current method for visualizing biochemical reaction networks is still incomplete in the sense that the actual rate at which a reaction is occurring not only depends on its rate constant but also on the concentrations of the reacting complexes. We will address this issue by incorporating simulation results into the network visualization. This obviously adds another level of complexity and the kind of information that will be visualized has to be selected carefully. The biologically relevant dynamical information will typically be at the level of patterns of states of molecular complexes or specific state sets and not on the structural level of complex species. Thus, displaying the complete dynamical state of a simulated model will be impractical and the viewer will have to dynamically select the most relevant aspect of information in a context-dependent way.

Currently, the NetworkViewer only displays reaction networks created with the Simmune Modeler. However, Simmune will soon be able to import rule-based models encoded in the upcoming SBML3 *multi* (multi-state, multi-component) standard[3]. At that point, the NetworkViewer can be used to visualize any rule-based model generated by approaches adhering to this standard.

[3]http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Multistate_and_Multicomponent_Species_(multi)

# Chapter 3: Visualization of Temporal Changes in High-dimensional Clinical Data

## 3.1 Introduction

Parkinson's disease (PD) is a chronic neuro-degenerative disorder characterized by gradual progression. Whereas motor impairments are the most recognized symptoms of PD, there is increasing recognition of the importance of a range of non-motor symptoms including cognitive decline, sleep disturbance, and depression [31]. For example, the presence of inter-relationships between clinical subgroups and disease progression is supported by the discovery that patients with greater postural instability and gait difficulty tend to have faster rates of progression and greater cognitive decline [32]. This paradigm shift has resulted in large datasets with diverse clinical and biologic markers [33]. For example, at the University of Maryland PD Center, we have collected 15 years of heterogeneous and multidimensional data (e.g. clinical and genomics) on $2,500$ PD patients across $20,000$ office visits.

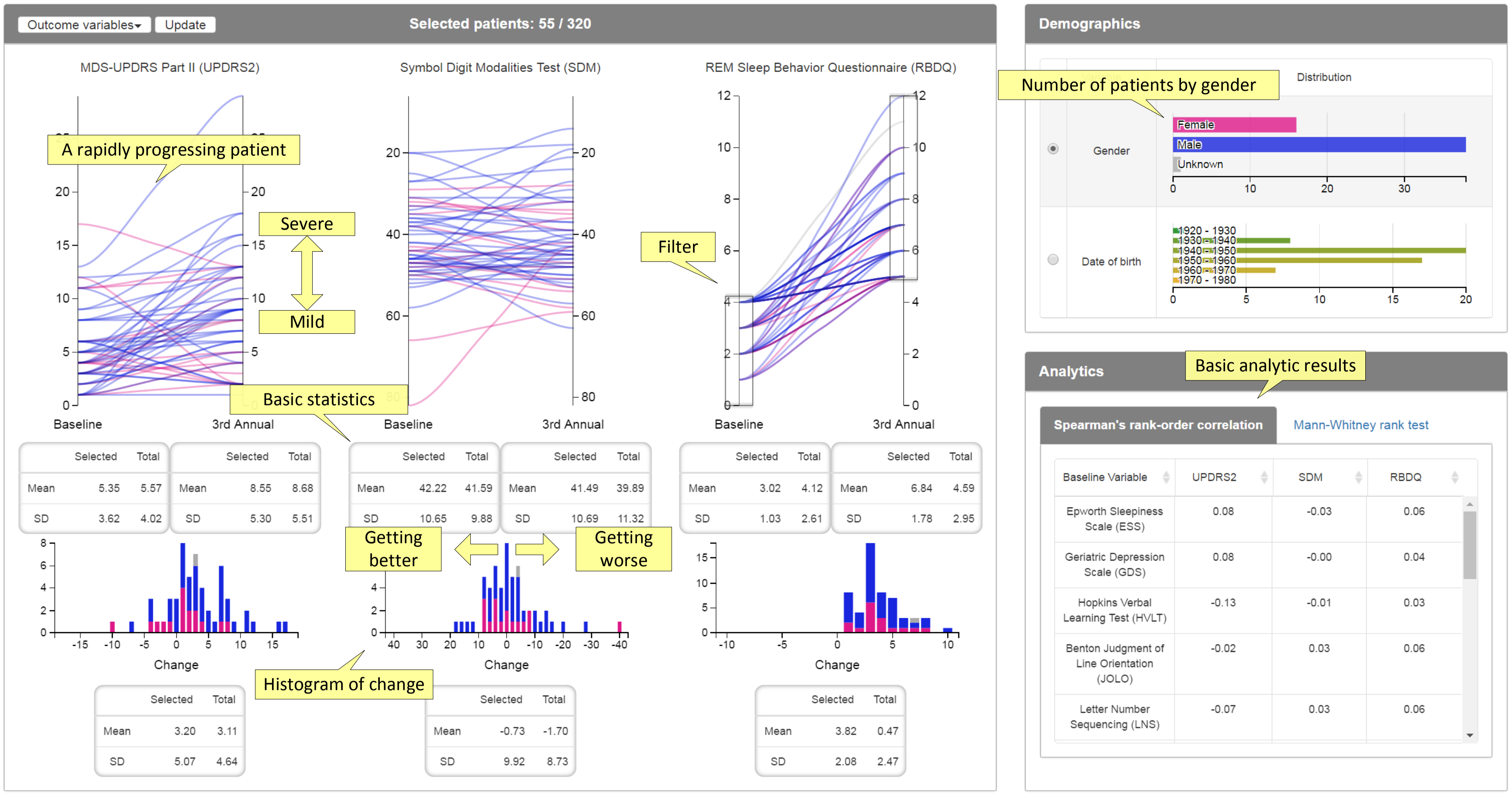
The expanded scope of disease beyond traditional motor symptoms raises the complexity of analysis significantly. These large multidimensional longitudinal datasets need new tools to identify disease subtypes and patterns of disease progression across diverse biomarkers, outcomes, and demographic subgroups. Conducting longitudinal studies of multidimensional data is challenging because of the limited human capacity to comprehend numerous interactions among a large number of variables. The immense cognitive load of such complex tasks causes these analyses to be time-consuming and ineffective. To address this challenge, many computational techniques such as data clustering and dimensionality reduction have been developed [34, 35]. Clustering techniques group similar data points (patients) into clusters based on specific criteria defined in a high-dimensional space. Dimension reduction techniques transform high-dimensional data points into low-dimensional representations that preserve crucial information required to proceed with the intended analysis.

Although these computational techniques provide a data-driven, objective perspective generated mathematically and statistically, they do not generate clinically meaningful results because clinical insight and experience with the domains is disconnected from the analysis. In addition, in some cases the stability of these results is questionable. For example, PD subtypes generated by cluster analysis were found to be inconsistent in various studies [36]. The results generated by pure data-driven approaches can therefore be misleading without proper clinical interpretation. We had a similar experience when we applied computational techniques in our preliminary studies, where the results were inaccessible to clinicians and not clinically relevant. This observation led us to believe that the most useful tools should be intuitive, interactive, and generate results interpretable by clinicians. This tool should allow clinicians to efficiently sift through arrays of longitudinal data, identify patterns, and generate hypotheses, which can then be subjected to rigorous statistical analysis.

In this chapter, we present a preliminary analysis of a public PD dataset with our tool Winnow (Figure 3.1) to study patterns of disease progression between time points. As a visual analytics tool, Winnow presents complex data in a way that facilitates detection of patterns and anomalies by the human eye. In contrast to the fully-automatic analysis conducted with computational techniques, visual analytics tools allow users to interactively explore the data through dynamic filtering and subgrouping. By interacting with intuitive visual representations, users can generate clinically relevant hypotheses and insights that are not easily accessible with conventional approaches.

Winnow visualizes multiple selected outcome measures simultaneously to enable the investigation of inter-relationships across outcome measures in various domains. By representing each patient as a line defined by the values of a selected outcome measure recorded at the first and second time points, the slope of lines indicates the rate of disease progression of the corresponding outcome. We also show a histogram of disease progression to facilitate the selection of fast- and slow-progressing patients. All visualizations are color-coded by demographic characteristics of patients (e.g. gender) to foster understanding of demographic-related questions such as whether female and male patients progress differently with respect to specific outcomes. The panels in Winnow are linked together to provide consistent visualizations during data exploration.

We have made our design choices in Winnow based on the feedback obtained



UPDRS2, Baseline:

| | Selected | Total |
|---|---|---|
| Mean | 5.35 | 5.57 |
| SD | 3.62 | 4.02 |

UPDRS2, 3rd Annual:

| | Selected | Total |
|---|---|---|
| Mean | 8.55 | 8.68 |
| SD | 5.30 | 5.51 |

UPDRS2, Change:

| | Selected | Total |
|---|---|---|
| Mean | 3.20 | 3.11 |
| SD | 5.07 | 4.64 |

SDM, Baseline:

| | Selected | Total |
|---|---|---|
| Mean | 42.22 | 41.59 |
| SD | 10.65 | 9.88 |

SDM, 3rd Annual:

| | Selected | Total |
|---|---|---|
| Mean | 41.49 | 39.89 |
| SD | 10.69 | 11.32 |

SDM, Change:

| | Selected | Total |
|---|---|---|
| Mean | -0.73 | -1.70 |
| SD | 9.92 | 8.73 |

RBDQ, Baseline:

| | Selected | Total |
|---|---|---|
| Mean | 3.02 | 4.12 |
| SD | 1.03 | 2.61 |

RBDQ, 3rd Annual:

| | Selected | Total |
|---|---|---|
| Mean | 6.84 | 4.59 |
| SD | 1.78 | 2.95 |

RBDQ, Change:

| | Selected | Total |
|---|---|---|
| Mean | 3.82 | 0.47 |
| SD | 2.08 | 2.47 |

Spearman's rank-order correlation

| Baseline Variable | UPDRS2 | SDM | RBDQ |
|---|---|---|---|
| Epworth Sleepiness Scale (ESS) | 0.08 | -0.03 | 0.06 |
| Geriatric Depression Scale (GDS) | 0.08 | -0.00 | 0.04 |
| Hopkins Verbal Learning Test (HVLT) | -0.13 | -0.01 | 0.03 |
| Benton Judgment of Line Orientation (JOLO) | -0.02 | 0.03 | 0.06 |
| Letter Number Sequencing (LNS) | -0.07 | 0.03 | 0.06 |

Figure 3.1: This screenshot of Winnow shows the results based on the Parkinson's Progression Markers Initiative (PPMI) dataset (Section 3.4). Winnow consists of three panels: the Outcomes panel (left), the Demographics panel (top right), and the analytics panel (bottom right). Here 55 patients (out of 320) with low baseline REM sleep behavior disorder (RBDQ $< 5$) and high RBDQ (RBDQ $\geq 5$) at the third annual visit are selected. The colors represent the gender of patients (magenta for females and blue for males).

from data scientists and clinicians in a long-term collaboration following the guidelines of Multi-dimensional In-depth Long-term Case studies (MILC) [37]. One example of the design choices we collaboratively made is reducing the complexity of visualization by choosing clinically relevant features, thus improving the usability of the tool. Clinical investigators want the opportunity for "hands-on" experience with clinical datasets to explore data based on experiential intuition. We repeatedly found the interpretability of results to be the top priority for the clinicians.

## 3.2 Related Work

### Data Clustering

Clustering techniques reveal patterns by grouping similar data points together. These techniques create another layer of abstraction on top of the raw high-dimensional data, thus allowing users to inspect the clustering results without directly addressing the high dimensionality of the data. Because the grouping of data points is based on an explicitly defined similarity function, defining a reasonable function is key to the success of clustering analysis. Typical $k$-means algorithm [38] defines dissimilarity of two high-dimensional points as the Euclidean distance between them. Density-based techniques such as OPTICS [34] rely on a neighborhood function to determine whether two data points are (similar) neighbors. Spectral clustering [39] requires a similarity function in constructing the adjacency graph. In practice, different similarity functions may lead to distinct results.

Nevertheless, the similarity of two data points is subjective, application-dependent, and usually difficult to define especially for high-dimensional data. This difficulty is fundamental because typical distance functions (e.g. Euclidean distance) fail to measure similarity precisely in high-dimensional space, a known problem referred to as the “curse of dimensionality”. In an exploratory clinical analysis that lacks a well-defined disease model, the limited interpretability of clustering results also aggravates the generation of clinically relevant, actionable results.

## Dimension Reduction

Dimension reduction techniques reduce data dimensionality while maintaining certain relationships among data points with respect to specific criteria. Principal component analysis (PCA) finds a series of mutually orthogonal principal components that account for the most variance in the data. Locally linear embedding (LLE) [40] and Laplacian Eigenmaps [35] first construct a neighborhood representation, typically based on pairwise Euclidean distances, and then derive a low-dimensional space that preserves local distances among neighboring data points. A neural network that learns to reconstruct the original high-dimensional training data from the derived low-dimensional latent variables can also be used for reducing data dimensionality [41].

All these techniques share similar limitations with clustering techniques such as requiring an explicitly-defined similarity function and limited interpretability of results. Whereas each principal component generated by PCA represents a linear combination of variables, the clinical meaning of adding (or subtracting) two clinically-unrelated features from different domains is unclear. The results generated by LLE and Laplacian Eigenmaps provide little clue for further interpretation. In fact, even extending a low-dimensional space derived by LLE or Laplacian Eigenmaps to include a new data point is non-trivial. The latent variables found by neural-network-based techniques are usually cryptic unless the training data are already labeled, which is not the case for exploratory analysis.

## High-dimensional Data Visualization

High-dimensional data visualization creates a visual abstraction of high-dimensional data such that patterns and anomalies can be detected by the human eye. Several successful visualizations of high-dimensional data such as the scatter plot matrix (SPLOM) [42] and parallel coordinates [43] have been widely applied. A SPLOM organizes scatter plots generated for each dimension pair as the elements inside a matrix; this layout facilitates efficient scanning of correlations between dimension pairs. A parallel coordinates visualization represents dimensions as individual parallel lines (axes); a data point is therefore represented as a segmented line connecting each point in order from the first axis to the last. Clusters of data points can therefore be visually detected as clusters of lines. Spreadsheet-based approaches that associate groups of cells with various types of visual representations can also help understand complex inter-relationships in heterogeneous datasets [44]. In contrast to the automatic analysis of computational approaches, data visualization can be easily combined with user interactions (e.g. brushing and linking) to allow interactive data exploration via filtering and zooming in and out of specific subsets of data.

The amount of information that can be displayed in a single visualization is, however, limited by the screen size and the perceptual capability of the human visual system. Such limitations can cause usability issues that adversely affect the efficacy of visual analytics tools. For example, visualizing all pairs of dimensions in a SPLOM is increasingly unrealistic as dimensionality grows. The parallel-coordinates visualization is incomprehensible even with a moderate number of dimensions. Better visual design strategies such as reordering dimensions [45], subsampling [46], and edge bundling [47] improve visual quality, but in general visualizing high-dimensional data remains a challenging task.

## 3.3 Preliminary Study

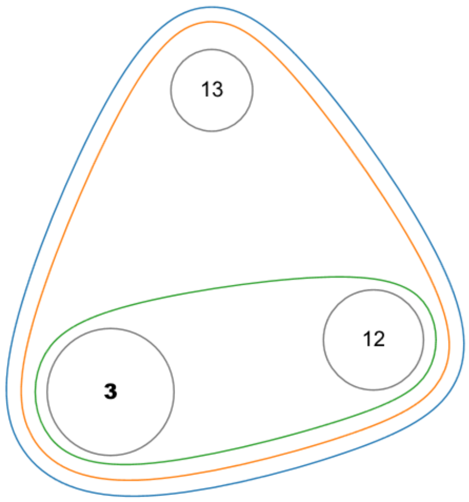
The Multi-dimensional In-depth Long-term Case studies (MILC) method [37] evaluates the efficacy of visualization tools using various approaches, such as interviews and observations, while collaborating closely with expert users. Here we describe the MILC we conducted as a preliminary study in the development of a clinician-friendly visual analytics tool.

We formed a multi-disciplinary research group of data scientists, database specialists, neurologists, and biostatisticians in 2015, targeting the analysis of the PD data collected at the University of Maryland PD Center. Group members included movement disorder specialists (neurologists) with extensive expertise in PD but little experience with visualization tools. In contrast, other group members with computational backgrounds were only familiar with computational and visualization techniques.

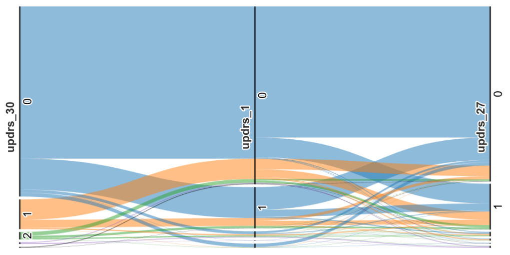
During each group session, a visualization tool or a new version of a tool based on previous group discussion was presented. Users then attempted to apply the tool for routine analysis tasks or confirming/rejecting PD-related hypotheses. The comments and feedback from users were recorded by the observers while they interacted with the tool.

Early in the project, we focused on identifying novel multi-domain PD subtypes to extend the recognized motor-based subtypes [48]. Our first attempt (April, 2015) was an interactive hierarchical clustering method based on OPTICS [34]. We implemented an interactive tool that enables users to explore alternative clustering results in a pre-generated hierarchy of clusters (Figure 3.2a). We then tried a series of dimension reduction techniques including PCA, LLE [40], Laplacian Eigenmaps [35], and t-Distributed Stochastic Neighbor Embedding [49] to reveal possible low-dimensional representations that lead to visually apparent clusters of patients.

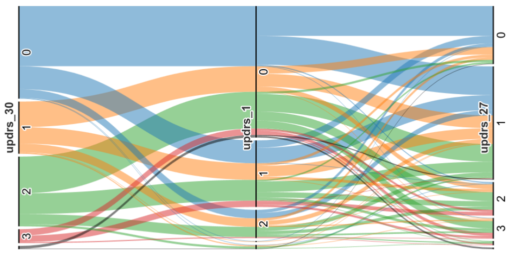
Common issues in all the above attempts were the limited interpretability of the results and the lack of flexibility for clinicians to steer and manipulate the analysis as desired. The goals from the clinician's perspective were to 1) visualize data on patient signs and symptoms in ways that align with their experience and 2) enable a simple hands-on interrogation of data to pose questions and generate hypotheses. We tried to improve user-friendliness by visualizing the constituent patient clusters but without success because of the complexity introduced by high



(a) Cluster hierarchy



(b) Cluster #3



(c) Cluster #12

Figure 3.2: (a) Our early attempt created a hierarchy of clusters to allow splitting and merging of clusters following a pre-generated hierarchy. Nevertheless, the parallel set visualization of two sibling clusters: cluster #3 (figure (b)) and cluster #12 (figure (c)), are difficult to interpret even with three dimensions.

data dimensionality.

For example, one of our early attempts created parallel sets visualizations for each cluster. The visualizations containing overlapping lines are, however, difficult to read even with only three dimensions (Figure 3.2b and 3.2c). The lack of flexibility is related to the limited capacity for user interventions in these techniques–the only way users can modify the results is through reconfiguring the parameters of the applied computational techniques. Furthermore, these two issues of interpretability and flexibility are related and may exacerbate each other. For example, one would not know how to modify the parameters of OPTICS without a proper interpretation of the results.

Acknowledging the obstacles in our group's process, we developed two key strategies: 1) switching to a simpler PD dataset and 2) organizing two half-day group retreats (July and August, 2016). We chose to use the Parkinson's Progression Markers Initiative (PPMI) dataset (Section 3.4) because it is a smaller, simpler, and more structured dataset with a well-defined protocol, thus allowing us to find patterns more easily with less interference from noise. Unlike the University of Maryland PD dataset, PPMI patients are more homogeneous (recently diagnosed and untreated at enrollment) and are assessed at more standardized intervals (every six months).

A consultant with expertise in applied biostatistics and modeling (Dr. Søren Bentzen) was invited to attend the retreats for a fresh perspective and to recharge the group dynamic. After a lively discussion, the major result of the first retreat was a preliminary sketch of Winnow, a clear breakthrough to achieve our goals. As compared to the previously developed methods, Winnow's data visualization is clinically intuitive and invites the hands-on experience that clinicians seek. In addition to Winnow's approach, we also discussed the following computational techniques for future extensions.

- Clustering analysis that groups patients by their temporal characteristics (e.g. multiple cross-sectional clustering or clustering the change in clinical features).
- Supervised learning methods that predict a user-defined rate of disease progression based on multiple outcome measures.
- Statistical methods, for example multivariate regression analysis, that prove or disprove hypotheses generated by users.

These techniques provide complementary strength to the visual analytics approach of Winnow. We also reached consensus to add a biomarkers panel in the future to include data from genomics, imaging, serology, and cerebrospinal fluid. Group members had mixed reactions on the importance of interactivity to a successful tool; although most members ranked interactivity high on the list of key components, some prioritized a robust data model or user-friendly interfaces over interactivity.

Based on the results of the first retreat, we designed an interactive visualization tool with the features found to be useful for clinicians. The second retreat was held in August, 2016 to present the first version of Winnow and to review its strengths and weaknesses. The group was unanimous in their positive assessment of the intuitiveness of the data visualization and the capacity for simple interaction with the data by users without intensive training (e.g. selecting patient subgroups and outcomes for analysis). We later developed new features proposed in the second retreat, including statistical tests for group comparisons. We describe the design of Winnow in Section 3.5.

## 3.4 Data

We present the following examples and case studies using data obtained from the Parkinson's Progression Markers Initiative (PPMI) database [50] (www.ppmi-info.org/data). The PPMI dataset contains patient and clinician-reported outcome measures as well as genetics, imaging and serologic data collected over a five-year period since 2010. As of now, more than 400 PD patients have been enrolled in the study. The PPMI is an ongoing study with patient records being updated on a rolling basis. Here, we use the data downloaded on March 2nd, 2017. For up-to-date information on the PPMI study, visit www.ppmi-info.org.

We selected the 16 outcome measures listed in Table 3.1 that were collected when patients were enrolled (BL) and at their third annual follow-up visit (Y3) to study disease progression. The MDS-UPDRS (Movement Disorders Society-Unified PD Rating Scale) measures PD severity and the other measures are selected from the autonomic, cognitive, sleep, behavior, and disability domains. We chose the third instead of the fifth annual visit to generate a representative set of data with a sufficient number of patients. We excluded patients with missing or incomplete data (e.g. enrolled for less than three years) from our analysis. In summary, a total of 320 patients (90 female, 211 male and 19 unknown) born between 1927 and 1979 were included. Each patient is represented by 34 features (16 outcome measures collected at BL and Y3, and two demographic attributes). The outcome measures are aggregated scores calculated from sets of items from validated questionnaires.

## 3.5 Visual Analytics Tool: Winnow

Figure 3.1 shows a screenshot of Winnow consisting of three panels: the outcomes panel (left), the demographics panel (top right), and the analytics panel (bottom right). The number near the top shows the total number of selected patients (55 in Figure 3.1). We also provide the option for descriptions of the selected ranges of the currently applied filters in a tooltip (not shown here).

Table 3.1: List of the 16 outcome measures and six domains of PD in our study.

| Domain | Outcome Measure | Description | Range |
|---|---|---|---|
| PD Severity: Motor & Non-Motor | MDS-UPDRS Part I (UPDRS1) | Non-motor aspects of experiences of daily living | 0–52 |
| | MDS-UPDRS Part II (UPDRS2) | Motor aspects of experiences of daily living (disability) | 0–52 |
| | MDS-UPDRS Part III (UPDRS3) | Motor examination for signs of PD | 0–132 |
| | MDS-UPDRS Total Score (T-UPDRS) | General PD severity: sum of MDS-UPDRS Part I-III | 0–236 |
| Autonomic | Autonomic Scale for Outcomes in PD (SCOPA-AUT) | Autonomic symptoms in PD | 0–75 |
| Cognitive | Hopkins Verbal Learning Test (HVLT) | Verbal short-term memory and new learning | 0–12 |
| | Benton Judgment of Line Orientation (JOLO) | Visuospatial perception and orientation | 0–30* |
| | Semantic Fluency (SFT) | Semantic and phonetic memory | $> 0$* |
| | Letter Number Sequencing (LNS) | Attention, working memory, and visuospatial ability | 0–21* |
| | Symbol Digit Modalities Test (SDM) | Processing speed | 0–110* |
| | Montreal Cognitive Assessment (MoCA) | Global cognitive evaluation | 0–30* |
| Sleep | Epworth Sleepiness Scale (ESS) | Daytime sleepiness | 0–24 |
| | Rapid Eye Movement Sleep (REM) Behavior Disorder Questionnaire (RBDQ) | Abnormal behaviors during REM sleep | 0–13 |
| Behavior | Geriatric Depression Scale (GDS) | Depressive symptoms in the elderly | 0–15 |
| | State-Trait Anxiety Inventory (STAI) | State and traits of anxiety in adults | 40–160 |
| Disability | Modified Schwab & England Activities of Daily Living (SEADL) | Independent and dependent function of daily activities | 0–100* |

* Lower score indicates more severe symptom. For all others, higher score indicates more severe symptoms.

### 3.5.1 Outcomes Panel

Each selected outcome measure corresponds to two plots in the outcomes panel: one showing the values at BL and Y3 (top) for each individual patient, and the other showing the number of patients with a certain amount of change between BL and Y3 as a stacked histogram (bottom). In the top plot, each patient is represented by a line connecting the values of that outcome measure at BL and Y3 for that patient. Using this visual representation, changes in values correspond to the slope of lines, whose differences can be detected efficiently by human eye [51]. We reverse the positive direction of axes in the plots, if needed, such that a positive slope always indicates PD progression (from mild to severe symptoms). Therefore, a line with a steep upward slope shows the corresponding patient is experiencing rapid progression on that outcome measure. This axis reversal is applied to Benton Judgment of Line Orientation (JOLO), Semantic Fluency (SFT), Letter Number Sequencing (LNS), Symbol Digit Modalities Test (SDM), Montreal Cognitive Assessment (MoCA), and Modified Schwab & England Activities of Daily Living (SEADL) where higher scores indicate milder symptoms (marked with an asterisk in Table 3.1).

Although the slope of lines can be easily seen for a moderate number of lines (patients), the difficulty in locating individual lines increases significantly with the degree of occlusion. We therefore also show the number of patients with a certain amount of change between BL and Y3 in a stacked histogram in the bottom plot to provide a summary of disease progression. A similar axis reversal is applied to the histograms such that a bar on the right represents more rapid progression. In both the top and bottom plots, the colors of lines and bars are determined by the selected demographic attribute in the demographics panel (Section 3.5.2).

Users can select a subset of patients by dragging the target interval on the target axis. During selection, all plots, including the ones in the demographics panel, are updated interactively to reflect the latest selected patients through brushing and linking. We show the mean and standard deviation of the values at BL, Y3, and the changes in between the two for the total sample and for the selected patients to allow for easy comparison.

For example, users can select the rapidly progressing patients whose UPDRS3 (motor exam) score increased by more than 12 over three years in the bottom histogram (bottom of Figure 3.3). After applying the filter, the 109 selected patients are represented by lines with steep upward slopes in the top plot (right of Figure 3.3). The mean UPDRS3 at Y3 for the selected patients (38.85, right arrow in Figure 3.3), is higher than the mean UPDRS3 at Y3 for the total patient sample (28.39, left arrow in Figure 3.3).

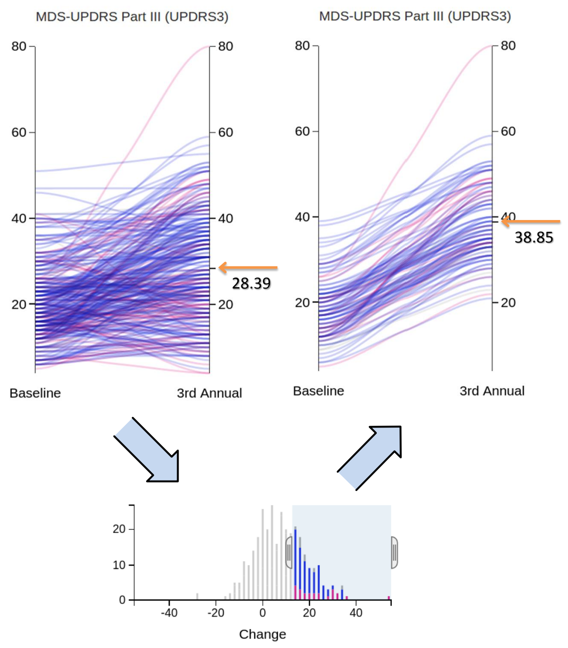


Figure 3.3: After using the bottom histogram for selecting patients with the most rapid progression in the motor domain ($\Delta$(UPDRS3) $> 12$), the corresponding plot of UPDRS3 in the outcomes panel shows only the 109 matching patients, represented by the lines with steep upward slopes. The mean UPDRS3 at Y3 of the group of selected patients (right arrow) is greater than that of the total patient sample (left arrow).

### 3.5.2 Demographics Panel

The demographics panel shows the gender and the year of birth in two individual histograms (top right of Figure 3.1). Instead of showing every year as a separate bar in the histogram, which would result in a crowded visual display, the years are grouped into decades to facilitate interpretation of age distribution and efficient use of the limited screen space.

The color scheme used in Winnow is determined by the selected demographic attribute. For example, when gender is selected, the lines (top plot) and bars (bottom plot) in the outcomes panel are colored in magenta and blue for female and male

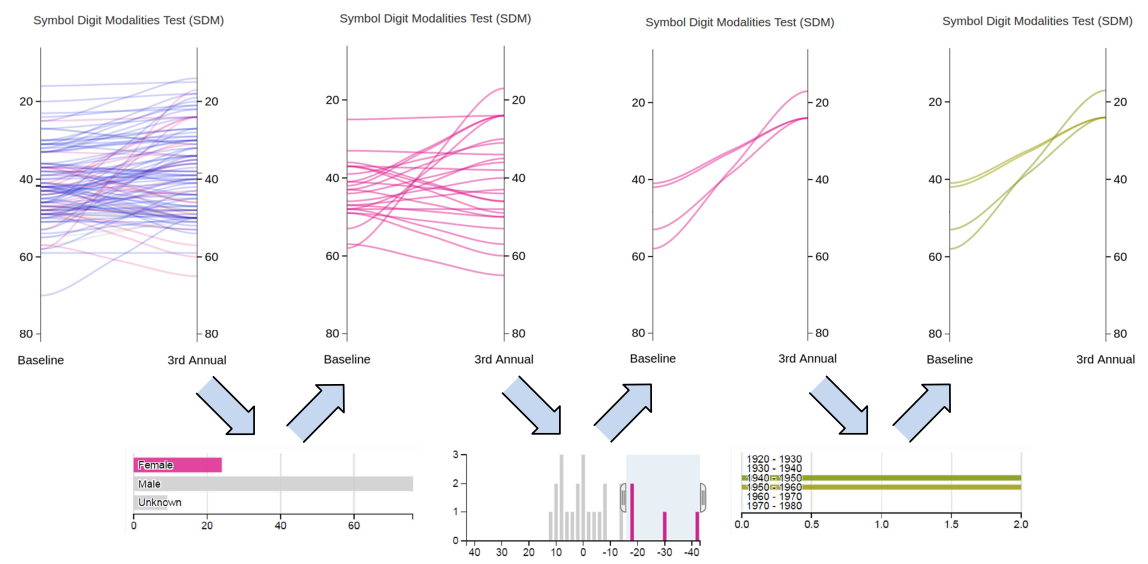


Figure 3.4: Users can filter the 109 previously selected patients (in Figure 3.3) by gender by clicking the corresponding bar in the demographics panel. After selecting female patients (left) with unusually rapid changes in cognitive function as measured by SDM (center), users can then change the color scheme to explore the year of birth of the selected patients (right).

patients, respectively. Users select a subset of patients with respect to a particular demographic attribute by clicking the corresponding bar.

For example, users can select the 24 female patients from the previously selected 109 rapidly progressing patients using the demographics panel (left of Figure 3.4). After applying the gender filter, the mean value of $\Delta$(SDM) is $-3.33$ (comparable to the decline of $-3.36$ before filtering by gender); this shows that the female patients in the selected cohort progress similarly to the mean total sample in terms of cognitive function as measured by SDM.

Users can also identify female patients with the most rapid progression in SDM. For example, they can select the four patients with $\Delta(\text{SDM}) < -16.62$, which is more than one standard deviation from the mean, represented by lines with significantly steeper slopes when compared with other female patients (center of Figure 3.4). If we select those four patients and switch from gender to year of birth in the demographics panel, we see that these four patients were born between 1940 and 1960, corresponding to ages between 56 to 76 years (right of Figure 3.4).

### 3.5.3 Analytics Panel

The analytics panel shows the relationships between pairs of variables through statistical analysis (bottom right of Figure 3.1). The first tab shows the correlation of a pair of outcomes evaluated by the Spearman's rank correlation; a high correlation coefficient for a pair of outcomes indicates that patients with severe symptoms in one outcome are likely to show severe symptoms in the other outcome; similarly, those with low on one outcome are likely to be low on the other. The second tab shows the *p*-values of the Mann-Whitney test comparing the distributions of outcomes in the selected patients and the remainder of the patients in the total sample; an outcome has a low *p*-value when its distributions are different in the selected patients and the remainder of the patients. Both the Spearman's rank correlation and the Mann-Whitney test are non-parametric. The variable tested can be selected from the values at BL, Y3, or the change in values from BL to Y3.

In the following example we use the Mann-Whitney test to compare the changes in values between two groups: The 109 fast-progressing patients with respect to UPDRS3 (selected in Figure 3.3) and the remainder of the patients ($n = 211$). The result shows, beside the trivial case comparing $\Delta$(UPDRS3) in the two groups, the other two UPDRS sub-scales ($\Delta$(UPDRS1) and $\Delta$(UPDRS2)–non-motor symptoms and disability) have low $p$-values ($p < 0.05$). Other variables with low $p$-values in increasing order are $\Delta$(GDS), $\Delta$(SDM), $\Delta$(SCOPA-AUT), $\Delta$(SEADL), $\Delta$(STAI), and $\Delta$(LNS), associated with a range of domains (behavior, cognitive, autonomic, and disability). In summary, disease progression in motor functions is associated with progression of autonomic dysfunction, cognitive decline, depression, anxiety, and disability.

### 3.5.4 Implementation

Winnow is implemented in JavaScript (frontend) and Python (backend). Users can conveniently run Winnow on modern browsers without installation through a link to the website[1].

## 3.6 Case Study

We now summarize two case studies we conducted with Winnow. The results shown in this section are for demonstration only. Analyses with Winnow are intended to uncover promising relationships between a range of outcome measures for selected patient subgroups. These analyses are for the purpose of generating hypotheses and need to be reproduced in future studies.

[1] http://hccheng.pythonanywhere.com/vis/

### 3.6.1 Questions

We have identified two clinically relevant questions about disease progression in PD with input from our clinical experts.

Q1 Does gender affect Parkinson's disease severity at year three?

Q2 Does the baseline severity of REM sleep behavior disorder (RBDQ) affect year three outcomes?

In the following we assess the effect of grouping using the Cohen's $d$ [52], which is the difference between two sample means divided by the pooled standard deviation:

$$d = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}}, \tag{3.1}$$

where $x_i$ and $\sigma_i$ are the mean and standard deviation of the $i$-th group. A Cohen's $d$ of value 0.2, 0.5, or 0.8 suggests a small, medium, or large magnitude of relationship, respectively [52]. We use Cohen's $d$ to provide a different but complementary perspective on group differences in addition to the rank-based Mann-Whitney test.

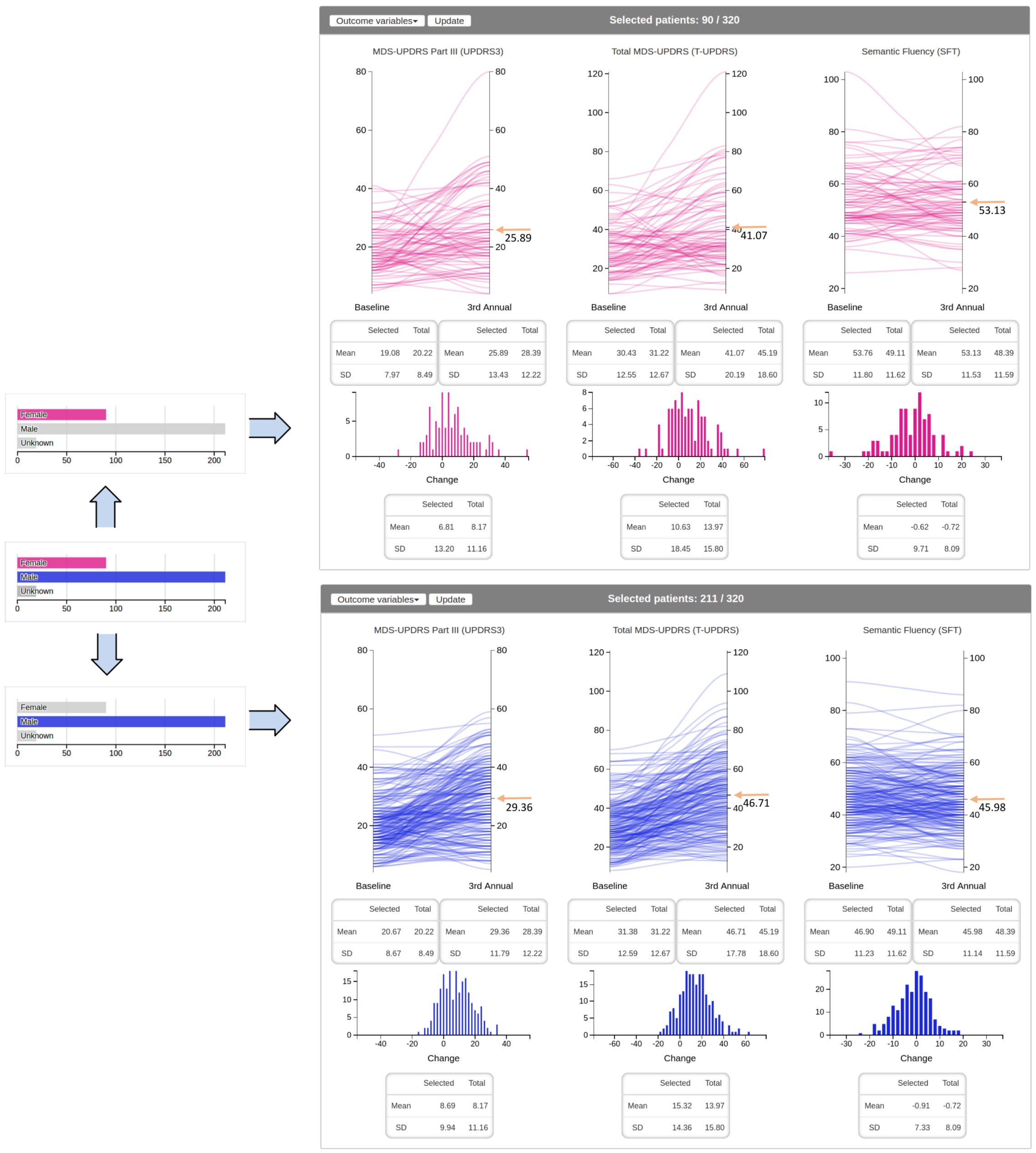
#### Q1: Does gender affect Parkinson's disease severity at year three?

Gender differences in PD-related symptoms have been studied in the past. For example, RBD (REM sleep behavior disorder) was shown to be more prevalent in male patients than female patients [53]. A thorough review of gender differences in cognitive functions can be found in [54]. In the following analysis, we first applied a filter in the demographics panel and then used the plots in the outcomes panel to investigate various outcome measures at Y3 (Figure 3.5).

For UPDRS3 in the motor domain, female patients have a mean value of 25.89 at Y3 (first plot in the top of Figure 3.5) and males have a mean value of 29.36 at Y3 (first plot in the bottom of Figure 3.5). Therefore, comparing female with male patients, females have less severe motor symptoms than males at Y3 ($d = 0.27$). The Mann-Whitney test shows a significant difference ($p = 0.003$, shown in Table 3.2) between the female and male patients. Other measures that significantly differ by gender at Y3 are listed in Table 3.2, including measures in the cognitive, sleep, and disability domains. Male patients show more severe symptoms than females in eight out of the nine outcome measures (marked in bold in Table 3.2). These seven outcomes are UPDRS2, UPDRS3, and T-UPDRS (general PD severity), SFT, MoCA and SDM (cognition), ESS (sleep), and SEADL (disability).

## Q2: Does the baseline severity of REM sleep behavior disorder (RBDQ) affect year three outcomes?

In this question, we investigate how the baseline severity of REM sleep behavior disorder (measured by RBDQ) affects the severity of outcomes in other domains over three years. We approached this question by selecting patients with increasingly severe REM sleep behavior disorder (greater values of RBDQ) at BL and comparing the severity of the other outcomes at Y3.



Selected patients: 90 / 320

MDS-UPDRS Part III (UPDRS3) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 19.08 | 20.22 |
| SD | 7.97 | 8.49 |

MDS-UPDRS Part III (UPDRS3) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 25.89 | 28.39 |
| SD | 13.43 | 12.22 |

MDS-UPDRS Part III (UPDRS3) — Change

| | Selected | Total |
|---|---|---|
| Mean | 6.81 | 8.17 |
| SD | 13.20 | 11.16 |

Total MDS-UPDRS (T-UPDRS) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 30.43 | 31.22 |
| SD | 12.55 | 12.67 |

Total MDS-UPDRS (T-UPDRS) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 41.07 | 45.19 |
| SD | 20.19 | 18.60 |

Total MDS-UPDRS (T-UPDRS) — Change

| | Selected | Total |
|---|---|---|
| Mean | 10.63 | 13.97 |
| SD | 18.45 | 15.80 |

Semantic Fluency (SFT) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 53.76 | 49.11 |
| SD | 11.80 | 11.62 |

Semantic Fluency (SFT) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 53.13 | 48.39 |
| SD | 11.53 | 11.59 |

Semantic Fluency (SFT) — Change

| | Selected | Total |
|---|---|---|
| Mean | -0.62 | -0.72 |
| SD | 9.71 | 8.09 |

Selected patients: 211 / 320

MDS-UPDRS Part III (UPDRS3) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 20.67 | 20.22 |
| SD | 8.67 | 8.49 |

MDS-UPDRS Part III (UPDRS3) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 29.36 | 28.39 |
| SD | 11.79 | 12.22 |

MDS-UPDRS Part III (UPDRS3) — Change

| | Selected | Total |
|---|---|---|
| Mean | 8.69 | 8.17 |
| SD | 9.94 | 11.16 |

Total MDS-UPDRS (T-UPDRS) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 31.38 | 31.22 |
| SD | 12.59 | 12.67 |

Total MDS-UPDRS (T-UPDRS) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 46.71 | 45.19 |
| SD | 17.78 | 18.60 |

Total MDS-UPDRS (T-UPDRS) — Change

| | Selected | Total |
|---|---|---|
| Mean | 15.32 | 13.97 |
| SD | 14.36 | 15.80 |

Semantic Fluency (SFT) — Baseline

| | Selected | Total |
|---|---|---|
| Mean | 46.90 | 49.11 |
| SD | 11.23 | 11.62 |

Semantic Fluency (SFT) — 3rd Annual

| | Selected | Total |
|---|---|---|
| Mean | 45.98 | 48.39 |
| SD | 11.14 | 11.59 |

Semantic Fluency (SFT) — Change

| | Selected | Total |
|---|---|---|
| Mean | -0.91 | -0.72 |
| SD | 7.33 | 8.09 |

Figure 3.5: After applying the gender filter, the plots show that female patients (top) have comparable motor symptoms (UPDRS3) and general PD severity (T-UPDRS) at BL and less severe symptoms at Y3 than male patients (bottom). For semantic fluency (SFT), female patients show less severe symptoms both at baseline and Y3. Arrows mark the mean values at Y3 for the males and females.

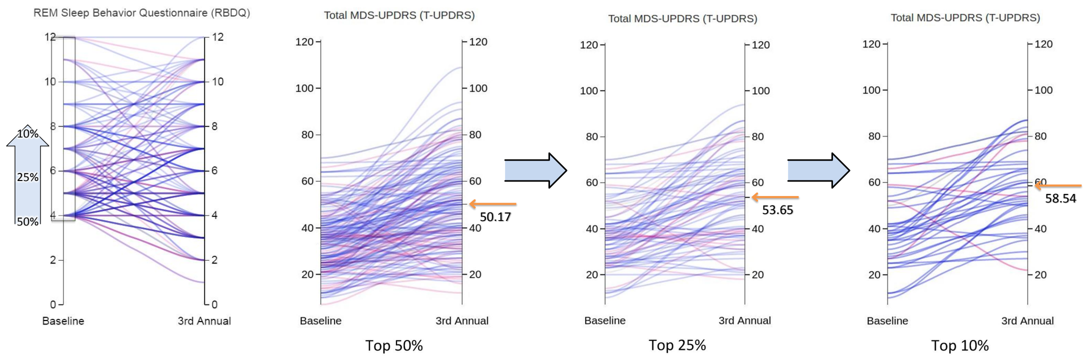
Table 3.2: Outcome measures that are significantly different ($p < 0.05$) by gender at year three (Y3)

| Domain | Outcome Measure | Mean value at Y3 |  | $p$-value |
|---|---|---|---|---|
|  |  | Female | male |  |
| PD Severity: Motor & Non-Motor | UPDRS2 | 6.98 | **9.31** | < 0.001 |
|  | T-UPDRS | 41.07 | **46.71** | 0.001 |
|  | UPDRS3 | 25.89 | **29.36** | 0.003 |
| Cognitive | SFT* | 53.13 | **45.98** | < 0.001 |
|  | JOLO* | **11.80** | 12.98 | < 0.001 |
|  | MoCA* | 27.08 | **25.94** | < 0.001 |
|  | SDM* | 42.48 | **38.19** | 0.006 |
| Sleep | ESS | 6.49 | **7.49** | 0.013 |
| Disability | SEADL* | 89.06 | **87.49** | 0.021 |

* Lower score indicates more severe symptom.

After applying a filter on baseline RBDQ with a cut-off value of four (Figure 3.6a), the mean values of T-UPDRS (PD severity) at Y3 is 50.17 for the top 50% of patients for RBDQ at BL (left of Figure 3.6b), and 39.96 for the bottom 50% of patients for RBDQ at BL (not shown here). These results show that for patients with greater baseline REM sleep behavior disorder, their general PD severity (T-UPDRS) is more severe at the third annual follow-up visit ($d = 0.57$).

In fact, if we use the analytics panel to compare the top and bottom 50% of baseline RBDQ ratings, the Mann-Whitney test shows that the distributions of nearly all outcome measures at Y3 are significantly different ($p < 0.05$) between the two groups except for a single cognitive measure, the semantic fluency test (SFT). These results show that the baseline severity of REM sleep behavior disorder is an important predictor of three-year outcomes across all the domains. Further studies



(a) Filtering baseline RBDQ

(b) Increasing mean T-UPDRS at Y3 from 50% to top 10% of patients (left to right)

Figure 3.6: (a) A filter that selects patients with increasingly greater baseline RBDQ. (b) As the filter moves to select patients with more severe baseline RBDQ (top 50%, 25%, and 10% from left to right), the mean T-UPDRS at Y3 increases from 50.17 to 53.65 and 58.54.

are needed to investigate this important finding.

The difference in outcomes at Y3 between patients with more and less baseline REM sleep behavior disorder becomes even more apparent if we adjust the filter to select increasingly greater RBDQ ratings at BL (Figure 3.6a). The patient groups with the top 50%, 25%, and 10% baseline RBDQ ratings have increasingly greater mean values of T-UPDRS at Y3 (50.17, 53.65, and 58.54 from left to right in Figure 3.6b). In summary, these results show a positive correlation between the severity of baseline REM sleep behavior disorder and the severity of multiple other domains at year three, indicating that the baseline severity of REM sleep behavior disorder is an important predictor of Parkinson's disease progression.

### 3.6.2 Discussion

Winnow enables clinicians and scientists to interactively and flexibly query complex and high-dimensional datasets with minimal training. This hands-on experience is critical for clinicians to generate hypotheses and to discover actionable, clinically-relevant results. Currently hypotheses are investigated with basic statistics; in the future we plan to use Winnow as an interface for complex analytic methods, such as supervised learning. For example, users can define patient subgroups with distinct profiles of progression in Winnow and then use these defined subgroups as the targets for supervised learning methods.

Based on the features of the dataset used in our examples, our analysis is limited to three years of longitudinal data. Whereas Winnow currently supports comparing two groups of patients (selected and unselected), future modification will enable selecting and switching between multiple patient groups to allow more sophisticated comparisons. In addition to the outcome measures used in our preliminary study, biologic data (e.g. genomics and serologic markers) may enable more precise and reliable groupings.

## 3.7 Conclusions and Future Work

Over the last 25 years, chronic medical conditions have been redefined with an expanding scope of symptoms and biologic disease markers. These advances have resulted in complex multidimensional clinical datasets. Such datasets pose substantial opportunities for discovery, but also pose unique challenges for traditional analysis. Parkinson's disease (PD) is a key example of these challenges–a complex progressive neurodegenerative disorder with rapidly expanding biologic data including genetics, imaging, and serologic markers. Because the diverse symptoms of PD progress over different timeframes, and are likely to vary in different patient subgroups, novel tools and approaches are needed to capitalize on the large, multidimensional, and longitudinal datasets available today.

In this chapter we present Winnow, a visual analytics tool that is intuitive, interactive and insightful for both scientists and clinicians. The design of Winnow is based on an intensive long-term collaboration between experts from diverse backgrounds. Winnow is a significant step to conduct complex analysis on existing databases that are necessary to advance the study of chronic medical conditions.

We are designing more concise and scalable visualizations by grouping similar patients in the outcomes panel and creating a graph-based representation summarizing the changes in outcomes [55]. We are also working on extending Winnow to incorporate new emerging biomarkers including genetics, imaging, serology, and biosensor metrics. The extension will also include a machine learning module to perform automatic data analysis.

# Chapter 4: Overview of Biomedical Images and Convolutional Neural Networks

## 4.1 Introduction

So far, we have presented our visual analytics tools for protein reaction networks and longitudinal multidimensional clinical data. In the rest of the dissertation, we focus on visualizations for biomedical images. Although human eyes can detect various features, from low level ones such as edges and corners to textures and even higher level concepts, inspecting images visually is both ineffective and inefficient. We therefore designed visualization and computational techniques for various biomedical image understanding applications.

In the following (Section 4.2), we first introduce the background for biomedical images and convolutional neural networks (CNNs), which is a specific type of neural network we used throughout the rest of this chapter. We present the unique characteristics of biomedical images and the challenges related specifically to their analysis, and the basics operations and terminologies in CNNs. We then present our studies on the visualization of gigapixel images (Chapter 5), time-lapse images (Chapter 6), and volumetric images (Chapter 7). Finally, we present a CNN-based segmentation for volumetric images (Chapter 8).

## 4.2 Background

### 4.2.1 Biomedical Images

Major breakthroughs in the pursuit of higher image resolution have enabled more accurate diagnoses and further understanding of biology. For example, modern cryo-electron microscopes capture images at high resolution on the scale of few nanometers to a couple of angstroms. At this resolution, researchers can now study biological structures at molecular scale, such as the ligand-protein complexes [56].

In the rest of the dissertation, we focus on three types of biomedical images: electron microscopy (EM) images, confocal fluorescence microscopy images, and magnetic resonance imaging (MRI). Depending on the imaging technology used, they can be further separated into sub-categories. For example, EM can be further categorized into transmission electron microscopy (TEM), which is based on transmitted electrons, and scanning electron microscopy (SEM), which is based on scattered electrons.

In contrast to natural images, many biomedical images are captured in 3D to enable the study of structural morphology and geometry of target objects such as brain tissues and subcellular organelles. For example, focus ion-beam scanning electron microscopy (FIB-SEM) generates high-resolution 3D volumes that have led to innovations in many biological systems [57]. Volumetric biomedical images are, however, usually anisotropic because of technical limitations: the resolution along the $z$-axis is usually much lower than that in the $xy$-plane (the imaging plane). When the discrepancy in resolution is insignificant, we can afford to reduce the resolution along the $xy$-plane to match that of the $z$-axis; when the $z$-axis resolution is orders of magnitude lower, we cannot apply such a reduction because it removes valuable high-frequency signals from the data.

Another significant difference between natural images and biomedical images lies in the number of channels. Typical natural images contain three color channels (i.e. red, green, and blue). Typical EM images are single-channel. For confocal fluorescence microscopy images, the channels depend on the dyes used (e.g. green fluorescent protein). MRI images usually contain multiple channels (e.g. T1 and T2) to differentiate various types of tissues.

In addition, the signal-to-noise ratio of biomedical images is significantly lower than that of natural images. The significant noise hampers precise depictions of objects, especially near object boundaries that are already ambiguous because of the irregular shapes of objects. Inconsistent illumination condition is also a prevalent and inevitable problem in biomedical images. Although denoising and normalizing early in the image processing pipeline can alleviate those problems, in practice designing robust computational methods for biomedical images remains challenging.

### 4.2.2 Convolutional Neural Networks (CNNs)

Recently, deep-learning-based methods, or convolutional neural networks (CNNs) specifically, have clinched top places in major computer vision competitions such as

ImageNet [58] for image classification and Pascal Visual Object Classes (VOC) [59] for semantic segmentation. The state-of-the-art methods for image recognition now achieve an accuracy comparable to humans; this remarkable achievement is a milestone in the pursuit of machine intelligence in the last few decades.

A typical feed-forward CNN consists of a sequence layers (from bottom to top), each performing specific operations. A layer is trainable if the weights (associated with neurons) in that layer are subjected to adjustment during training. Convolutional and fully-connected layer are two most common types of trainable layers. In a convolutional layer, a kernel is a trainable tensor that defines the filtering weights. For example, a $2D$ convolutional layer slides $k$ kernels (i.e. $W_{m,1}, W_{m,2}, \cdots, W_{m,k}$) over the input 3D tensor $U$ of size $x \times y \times m$, where $x$ and $y$ correspond to the spatial dimensions, and $m$ denotes the number of input feature maps (or, channels). After filtering the input tenser $U$ with the kernels $W_{m,k}$, the layer outputs a 3D tensor $V$, which is connected to the next layer. The $i$-th output feature map $V_i$ is calculated as $V_i = \sum_m W_{m,i} * U_m + b_i$, where $b_i$ denotes the associated bias term. The filtering can be selectively applied to neurons separated by a predefined distance referred to as stride. In contrast to convolutional layers that connect neurons in a local neighborhood, a fully-connected layer calculates the value of an output neuron as a weighted sum of all the input neurons; the fully-connected layers are also used in conventional multilayer perceptrons.

A nonlinear activation function converts the values in an output feature map $V_i$ into corresponding neuron responses; such an activation mechanism simulates artificially the activation of neuron cells and injects nonlinearity into the model.

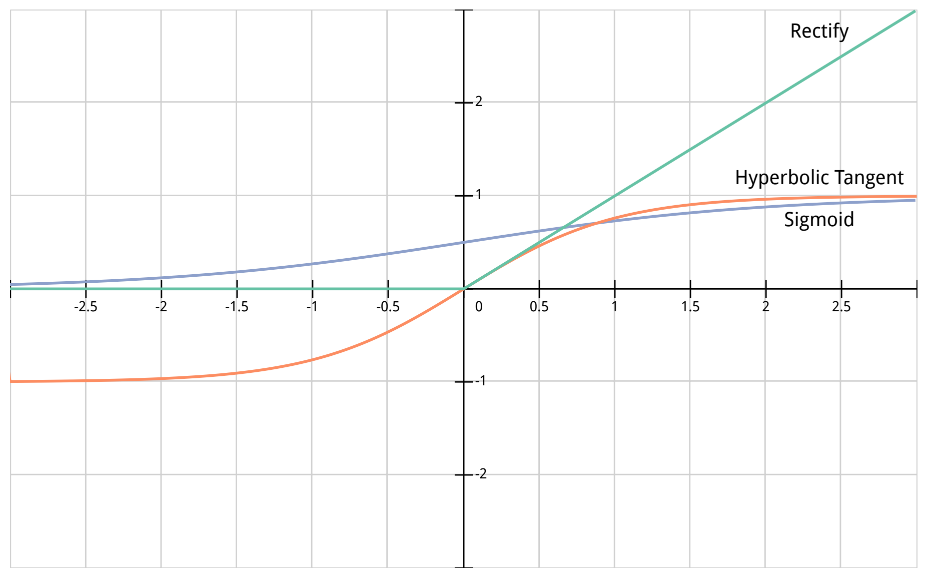


Figure 4.1: Three nonlinear activation functions: sigmoid (blue), hyperbolic tangent (orange), and rectify (green).

Common activation functions are sigmoid, hyperbolic tangent, and rectify (Figure 4.1). Different activation functions are preferred depending on the targeting output distribution. For example, the hyperbolic tangent function with a range of $(0, 1)$ can be used when we want the output to represent a probability value.

Non-trainable layers, such as spatial pooling layers, always perform the same predefined operation. For example, a max-pooling layer always outputs the maximum values within a local neighborhood by filtering the input tensor with a maximum filter. Usually, a pooling layer is used to reduce the spatial dimension in subsequent layers by setting a stride larger than one. Pooling layers also add more nonlinearity to the network if the operation used is nonlinear (e.g. maximum or minimum).

During training, a solver adjusts the trainable parameters in a CNN by a backward pass (i.e. backpropagation), which follows a forward pass of training samples from the bottom layer to the top layer; the adjustment is based on gradients calculated from a user-defined loss function. Depending on the application, common loss functions include binary cross-entropy and mean squared error.

Though CNN architecture varies from one design to another, most common networks resemble that of AlexNet [60]. AlexNet has seven trainable layers, including five convolution layers and two fully-connected layers. Three max-pooling layers progressively reduce the spatial resolution of deeper layers (i.e. layers near the top). Current research in deep learning is towards building deeper and deeper networks for increasingly complex problems. Nevertheless, increasing the number of layers (and also the number of trainable parameters) can deteriorate performance. This deterioration is related to the vanishing gradient problem, which challenges the tuning of deep layers, or the overfitting problem, where the model simply learns the input samples.

## CNNs for Biomedical Images

As a general machine learning technique, CNN has the potential multidisciplinary impact beyond natural images. For example, deep learning may improve the accuracy of the clinical diagnoses provided by computer-aided diagnosis systems in recognizing tumorous tissues in mammography images.

In fact, an encouraging trend has started to emerge in the biomedical image analysis community. For example, in three consecutive years from 2014 to 2016, an increasing number of participants (i.e. three, five and nine) have applied deep-learning-based methods in the Brain Tumor Image Segmentation Challenge (BRATS) [61]. Another example is the MRBrainS segmentation challenge [62], in which challenge leaders switch their choice of classifier from Random Forest and Support Vector Machine (SVM) (blue and green in Figure 4.2, respectively) to deep-learning-based techniques (orange in Figure 4.2) in 2016.

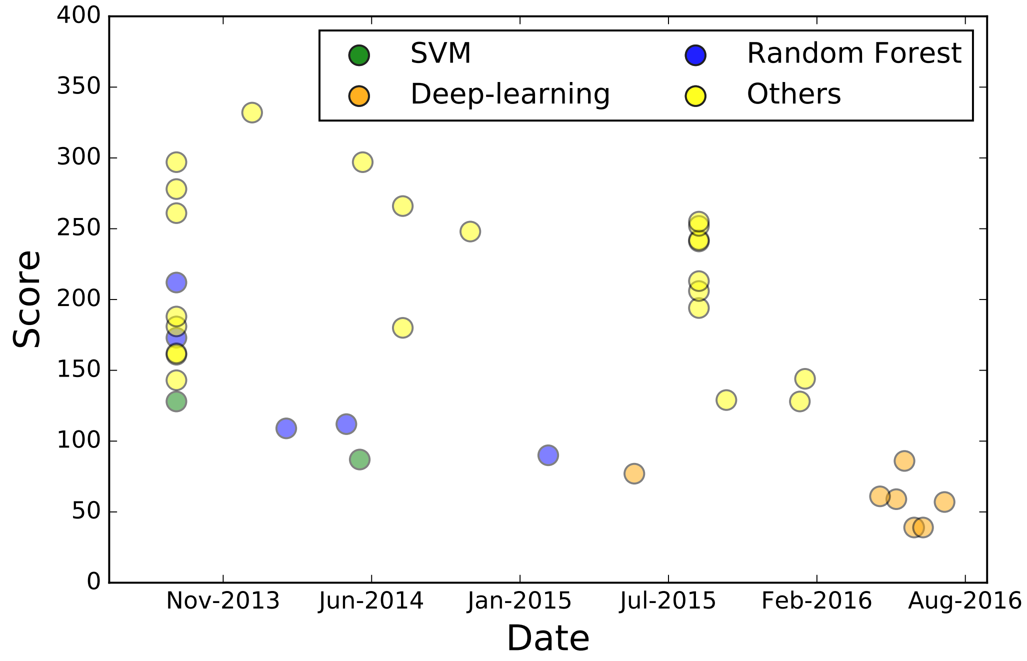


Figure 4.2: The techniques used in MRBrainS segmentation challenge shift from Random Forest (blue) and SVM (green) to deep-learning-based techniques (orange) from 2013 to 2016. Lower scores indicate higher performance.

The general guidelines in designing CNNs, such as alleviating the vanishing gradient problem, speeding up convergence, and multiscale analysis, are independent of applications. For example, several recent studies on biomedical image segmentation [63, 64, 65] have reported superior performance by combining fully convolutional networks [66] with residual networks [67]; both techniques were developed originally for natural images. Nevertheless, biomedical images are different from natural images in many aspects (e.g. anisotropicity and low signal-to-noise ratio, c.f. Section 4.2.1). In addition, annotated biomedical datasets are scarce and relatively small in size if compared with natural image datasets. Different designs of CNN might be needed to address these application-specific challenges.

# Chapter 5: Visualization of Microstructures in Gigapixel Microscopy Images

## 5.1 Introduction

Advances in high-resolution imaging technology allow researchers to see fine details of objects. As technology pushes the limits of image resolution, the widening gap between image and screen resolutions has brought new challenges for image analysts. For example, visually inspecting a 281-gigapixel microscopy image [5] on an ordinary four-megapixel screen would require repeatedly zooming in and out of the image. This routine is especially tedious at the beginning of exploring the image when analysts have yet to decide which specific targets to focus on. The analysis of gigapixel images therefore requires intense human-computer collaboration, especially in non-trivial analyses that are laborious even for moderate-sized images.

Many examples of such demanding analyses involve texture detection, which is paricularly challenging because of the complex nature of textures. Besides the complexity in modeling textures computationally [68], visually inspecting textures in large images is also difficult because different textures can easily become indistinguishable when high-frequency signals are removed after downsampling. Analysts are therefore faced with two conflicting operations: 1) zooming in to specific regions of the image for the high-resolution details required in texture analysis, and 2) zooming out for a global view that puts the target microstructures in a better context [69, 70].

Although the conflicts caused by resolution disparity aggravate image understanding, additional visual cues such as color [69] and depth [71] can enhance human perception of images. In the past, specialized lighting functions [72] and clustering techniques [73] have been used to assign colors and opacities that highlight brain microstructures in volumetric data. Studies in visual psychophysics have also revealed that human eyes are naturally more efficient in processing colors than textures [51]. Therefore, in this work, we use colors to highlight texture differences and facilitate the search for distinct textures and microstructures in large images. Whereas the resolution disparity problem is inevitable when exploring large images, texture differences that are less apparent at a given resolution stand out more easily because the color-based highlights are more salient to human eyes

We address the following challenges. First, conducting texture analysis on gigapixel images is extremely computationally expensive. Existing segmentation algorithms that model an image as a grid graph (e.g. F&H [3]) or a feature matrix (e.g. FSEG [4]) can handle only small images because of the excessive memory usage (circles and crosses in Figure 5.1). Second, the textures of interest are not discernible at all scales. The need for finding and visualizing texture differences across scales in large gigapixel images calls for an efficient segmentation procedure and a flexible interaction component to enable manual exploration of gigapixel images.

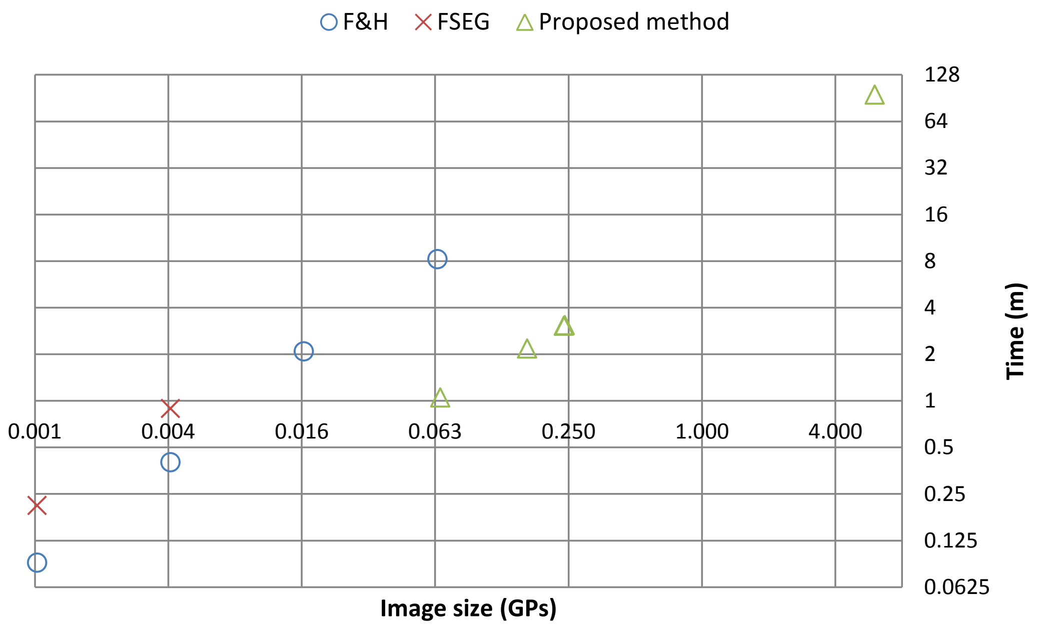


Figure 5.1: The running time (m) of our hierarchical segmentation algorithm is considerably lower than F&H [3] and FSEG [4]. Both image size ($x$-axis) and time ($y$-axis) are in log scale. For large images, F&H and FSEG terminated prematurely because of insufficient memory.

We design our gigapixel-scale segmentation algorithm that addresses the two challenges (triangles in Figure 5.1). We apply an efficient hierarchical segmentation algorithm to identify regions with different textures at various scales (Section 5.4) based on the joint distribution of intensity and noise-resistant local binary pattern (Section 5.3). Similar to previous research on gigapixel images [69, 74, 75], we use a Gaussian pyramid as the multiscale image representation. A tile-based storage divides each pyramid level into a set of rectangular tiles of size $1024 \times 1024$, allowing efficient loading of relevant regions in the image pyramid at various scales. In addition, we apply a superpixel segmentation and model the image as a less memory-demanding graph of superpixels. Our hierarchical texture-based segmentation identifies regions that differ in texture, starting from the most downsampled, lowest-resolution level all the way to the base, full-resolution level in the pyramid.

An analogy of such coarse-to-fine process can be found in human response to visual stimuli, in which coarse information enters the system first, providing cursory interpretation, which is then followed by processing specific, detailed information [76].

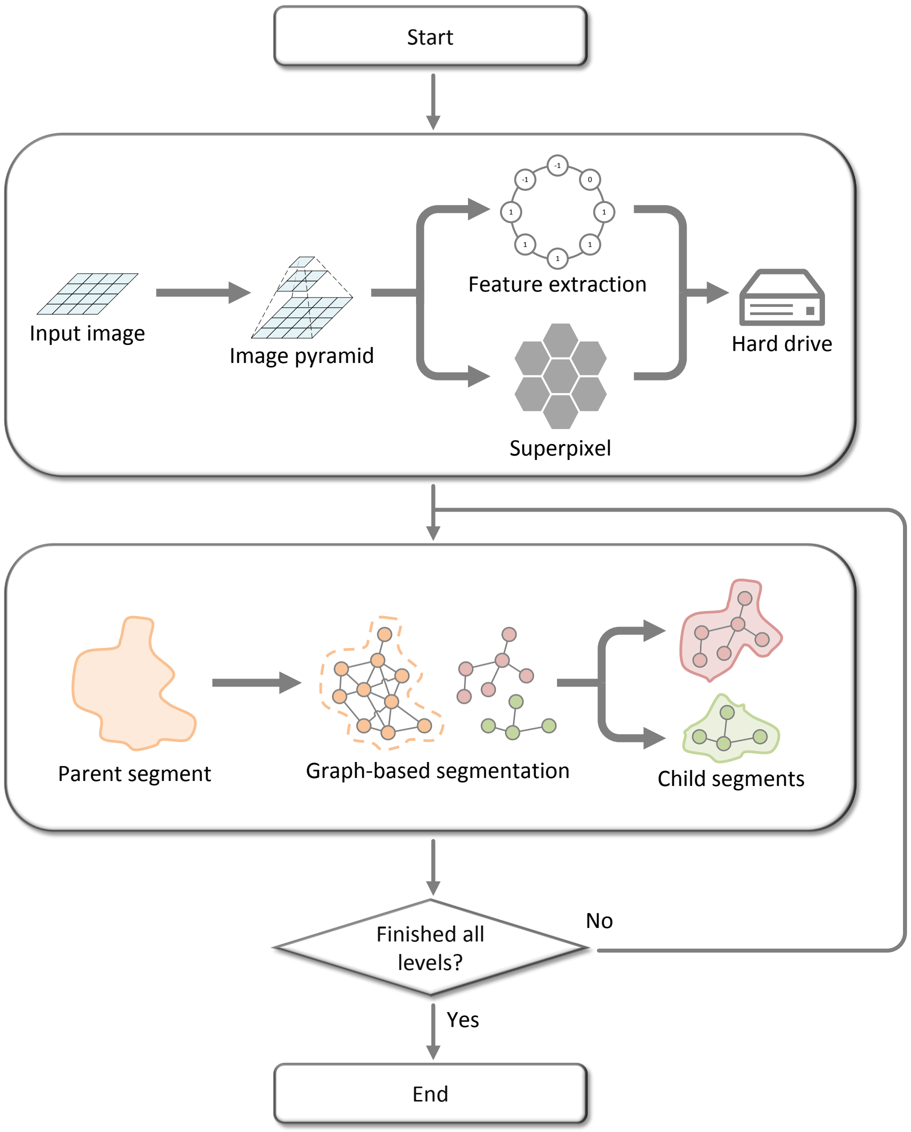


Figure 5.2: An overview of the segment tree building procedure. Noise-resistant local binary patterns and superpixels are generated for each tile at all pyramid levels. A hierarchical graph-based segmentation subdivides each parent segment into child segments.

At each level of the hierarchical segmentation, we use an unsupervised graph-based segmentation procedure to subdivide all segments into child segments. Figure 5.2 shows an overview of the system, which includes the image pyramid building, the feature and superpixels generation, and the hierarchical segmentation. The hierarchy formed by segments across different scales can be represented as a segment tree, which facilitates the multiscale exploration of alternative segmentations interactively. During user interaction, segments are colored differently to allow visual detection even when zooming out. Based on the color-based highlights, users can decide whether they need to split a segment further (Section 5.5). Such a user-mediated approach may significantly reduce the time needed to locate regions with texture differences, reveal ones that could otherwise be overlooked, and lead to a more thorough exploration.

## 5.2 Related Work

### 5.2.1 Texture Analysis

Texture analysis has received much attention in image processing and computer vision, yet it remains a difficult problem because of its complexity. Tuceryan and Jain grouped texture analysis methods into four categories: statistical, geometrical, model-based, and signal processing-based methods [68]. Statistical methods describe texture using statistics derived from the intensity distribution in a local neighborhood. For example, Haralick features [77] are statistics (e.g., correlation and entropy) calculated from co-occurrence matrices. Geometrical methods describe texture as a spatial distribution of texture primitives, identified as homogeneous regions [78] or edges [79]. Model-based methods create generative image models, such

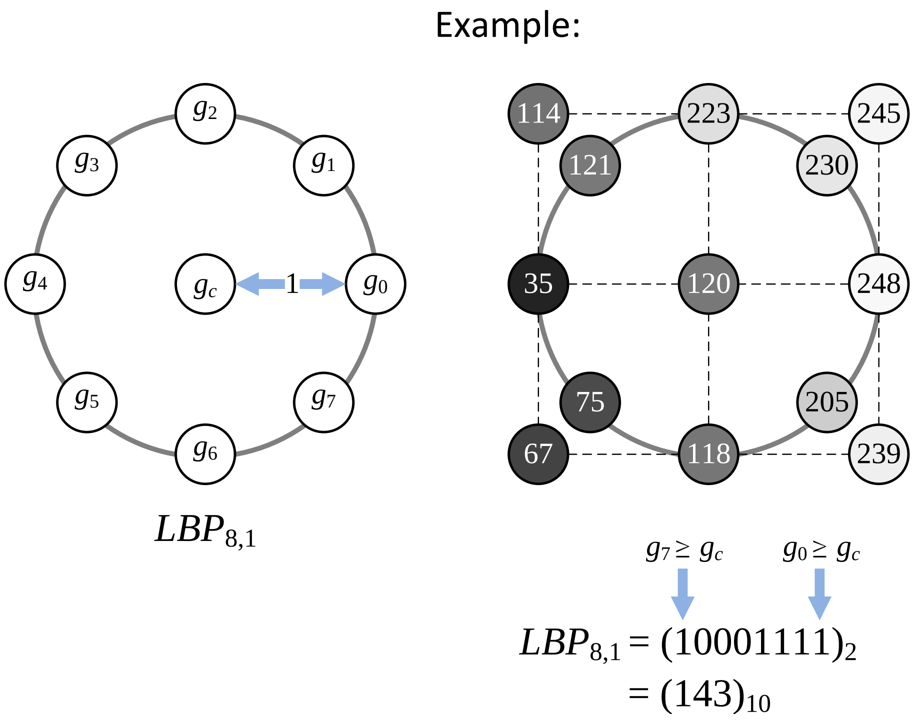


Figure 5.3: This example shows the calculation of $LBP_{8,1}$ using eight sample values, $g_1$ to $g_8$, taken on a unit circle (left). With the values shown in the right hand side, the corresponding LBP is a binary string `10001111`, which is 143 in decimal (right).

as fractal model [80], that can synthesize the observed intensity distribution. Signal processing-based methods perform analysis after extracting features from filtered images [81, 82, 83].

Inspired in part by Haralick features, the local binary pattern (LBP) method [84, 85] labels a pixel based on the relationships between the intensity value of that pixel and the intensity values of its neighbors. These local patterns serve as low-level texture primitives that, as opposed to the high-level primitives such as edges, contain local information at pixel level.

An LBP, denoted by $LBP_{P,R}$, is represented as a binary string of length $P$, in which each bit encodes the relationship between the intensity value of the center pixel and the intensity value of the neighbors, which are $R$ pixels away from the center. When a neighboring pixel has value greater than or equal to that of the center pixel, the corresponding bit is one (otherwise, it is zero). Figure 5.3 shows an example of the calculation of $LBP_{8,1}$. With $P$ equals eight, the binary string is eight bit long, representing 256 possible patterns. A 256-bin LBP histogram summarizing the frequency of patterns in a region describes the texture of that region. The distance measure of two LBP histograms therefore represents the texture dissimilarity between two corresponding regions.

A number of variants have been proposed to improve the classic LBP method. Instead of using a predefined set of "uniform" patterns [85], the dominant (i.e. most frequent) patterns contain more textural information [86]. Magnitude components can provide complementary information to the original LBP, thus improving the capability in discerning subtle patterns [87]. Local ternary pattern (LTP) is more resistant to noise after adding a third state to explicitly represent uncertainty caused by noise [88]. Noise-resistant local binary pattern (NRLBP) maps the third (uncertainty) state in LTP into candidate uncertainty-free patterns and use soft histograms to accumulate the occurrences of candidate patterns after mapping [89].

### 5.2.2 Image Segmentation

Supervised image segmentation extracts knowledge (e.g. property of classes of tissue [90]) from labeled samples in the training phase and then generates segmentation in the testing phase based on the training feedback. Having high-quality training data, therefore, is a precondition for a successful segmentation. The need for training samples, however, limits supervised approaches to applications with readily available samples, which is uncommon because labeling is tedious and labor-intensive.

Unsupervised segmentation, on the contrary, detect boundaries directly based on distribution of pixels in target images. The lack of a priori information, however, adds another layer of difficulty. Conventional unsupervised clustering methods such as $k$-means [91] or fuzzy $c$-means [92] can be adapted to solve segmentation problems.

A number of graph-based methods formulate the image segmentation problem as a graph cut problem, minimizing cost functions such as normalized cut [39] and min-max cut [93]. In general, finding these graph cuts requires solving hard optimization problems (e.g., NP-complete or NP-hard) and therefore restricts their use to small graphs (or images) because of the lengthy computation time. Felzenszwalb and Huttenlocher propose an efficient segmentation based on the minimum spanning tree of the graph representing an image [3]. In practice, the method runs in near-linear time with respect to the number of pixels. Such efficiency enables segmentation at video rate and segmentation of larger images. However, as we discuss earlier (cf. Figure 5.1), this algorithm does not scale up to gigapixel images.

Because many image processing tasks are complex, they become easily intractable for large images. Superpixel algorithms generate an initial segmentation of the image by grouping pixels into regions called superpixels to simplify subsequent operations. Superpixel algorithms are therefore often used as a preprocessing step for computationally demanding tasks [94, 95, 96]. Targeting efficient generation of compact regular-shaped superpixels, the simple linear iterative clustering

(SLIC) [97] method uses a variant of $k$-means clustering to generate high-quality results compared with state-of-the-art algorithms.

Because segmenting images is computationally demanding, we cannot directly apply existing algorithms to large gigapixel images. By segmenting progressively from low to high resolution levels in the scale-space pyramid as well as using the superpixel representation, we reduce the required memory to a manageable size and carry out large-scale image segmentation.

## 5.3 Joint Intensity-texture Histogram

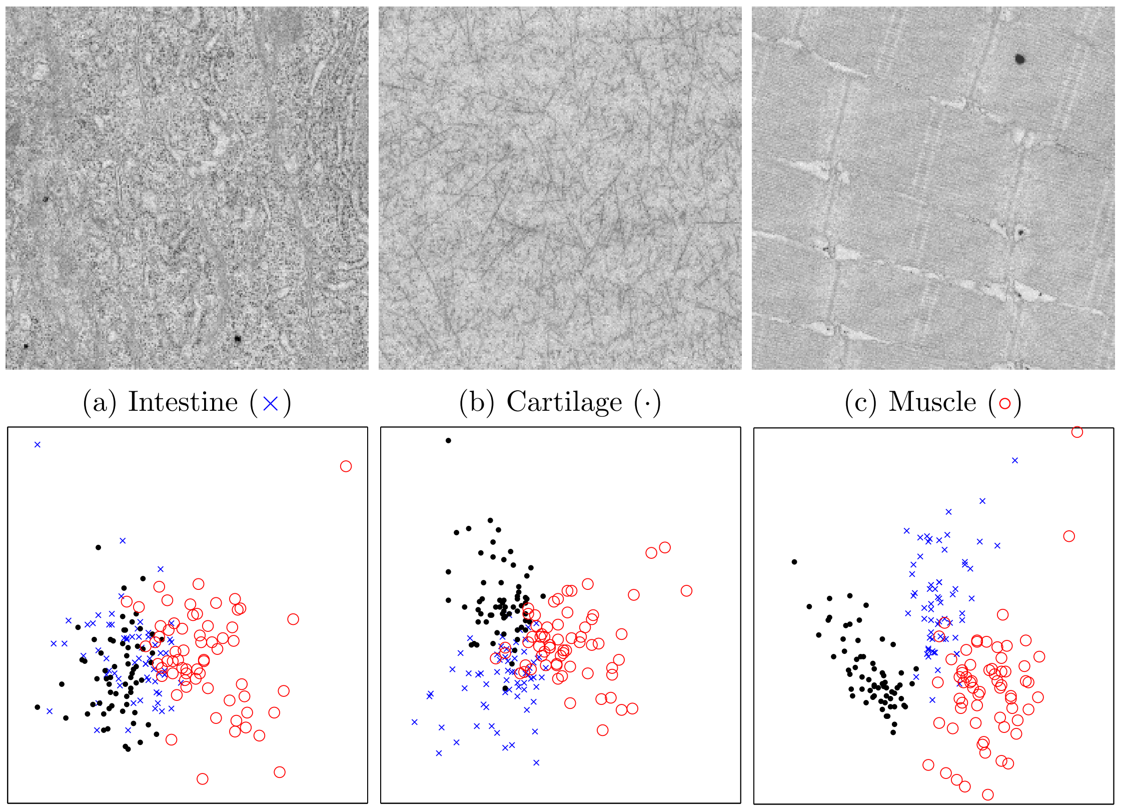
We use the joint histogram of intensity and noise-resistant local binary pattern (NRLBP) [89], which is a variant of local binary pattern (LBP) [84, 85], to identify texture differences. An LBP, $LBP(P, R)$, is a binary string of length $P$ in which each bit encodes the sign of the difference between the intensity value of the center pixel and the intensity value of a neighboring pixel $R$ pixels away from the center. Nevertheless, this binary encoding is prone to noise that causes unstable sign of intensity differences. Local ternary pattern (LTP) [88] alleviates the sensitivity to noise by a ternary encoding that uses the third state to represent situations when the intensity difference is less than a predefined threshold. NRLBP [89] treats the third state of LTP as a wild card that can map to either zero or one. Each ternary encoding of LTP therefore corresponds to a specific set of conventional uniform LBPs [85] after mapping the wild cards to zero or one.

The LBP-based features describe textures based on intensity differences instead of actual intensity values. Nevertheless, intensity values are important attributes in many applications, including the analysis of microscopy images in this work because tissues of the same type may fall in a specific range of intensity values. By combining intensity with NRLBP, we use the joint distribution of intensity and NRLBP to improve the descriptive power. The joint histogram divides the intensity range $[0, 255]$ into eight non-overlapping bins and uses a separate bin for each pattern of NRLBP. We use the chi-square distance $\chi^2(x, y)$ to compare two normalized intensity-NRLBP joint histograms $x$ and $y$; $\chi^2(x, y)$ is defined as:

$$\chi^2(x, y) = \sum_{i,j} \frac{(x_{i,j} - y_{i,j})^2}{x_{i,j} + y_{i,j}}, \tag{5.1}$$

where $x_{i,j}$ and $y_{i,j}$ are the values of bin $(i,\ j)$ in $x$ and $y$, respectively.

Figure 5.4 shows the power of the intensity-NRLBP joint histograms by visualizing the result of multidimensional scaling (MDS) [98] with the intestine (Figure 5.4a), cartilage (Figure 5.4b), and muscle (Figure 5.4c) tissue samples. Each sample of size $256 \times 256$ is divided into $8 \times 8$ blocks (each of size $32 \times 32$), which are projected to a two-dimensional space based on the pairwise chi-square distances computed using LBP (Figure 5.4d), NRLBP (Figure 5.4e), and the joint distribution of intensity and NRLBP (Figure 5.4f). By modeling noise explicitly, NRLBP separates the blocks of different tissue types better than LBP. Using the joint distribution of intensity and NRLBP further separates the blocks into three distinguishable clusters, one for each tissue type. Throughout this work, we use the NRLBP based on $LBP_{8,1}$.


(a) Intestine ($\times$) (b) Cartilage ($\cdot$) (c) Muscle ($\circ$)

(d) LBP (e) NRLBP (f) Intensity+NRLBP

Figure 5.4: The two-dimensional projections (d–f) show the distribution of sample blocks of (a) intestine, (b) cartilage, and (c) muscle tissues. The projections are obtained by MDS analysis based on the pairwise distances computed using (d) LBP, (e) NRLBP, and (f) joint distribution of intensity and NRLBP.

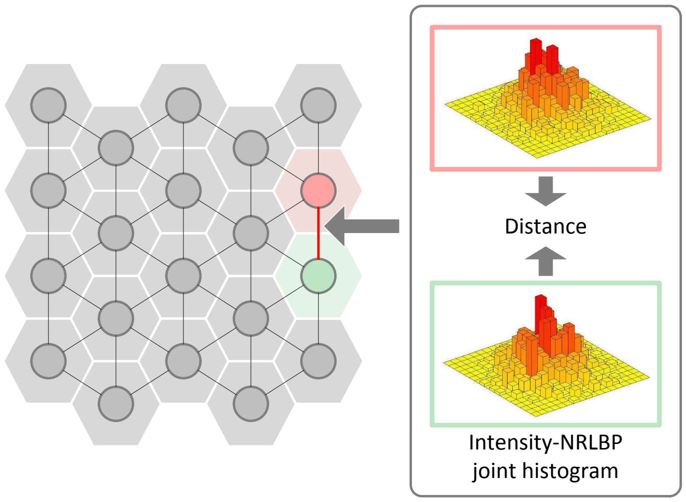
## 5.4 Graph-based Multiscale Segmentation

For large images, working directly at the pixel level is prohibitive because of the computational burden. Instead, we perform the actual texture segmentation at the superpixel level. Because the number of superpixels is two orders of magnitude smaller than the number of pixels, the computational burden becomes less of an issue.

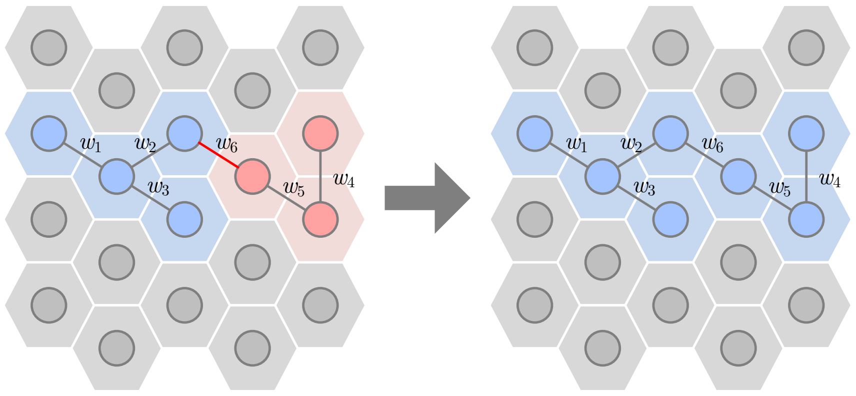
We use the simple linear iterative clustering (SLIC) [97] to efficiently segment each image tile into superpixels of size about 500 pixels. The SLIC generates regular-shaped superpixels and therefore avoids the bias introduced by the irregular distribution of nodes in a graph, which may diminish the quality of graph-based segmentation techniques [96]. We set the compactness to ten to generate results with a balanced shape regularity and boundary accuracy. The superpixel extraction is repeated for each pyramid level independently.

Given the superpixels, we first construct a superpixel adjacency graph $G$ and then partition $G$ into disjoint connected components. Each connected component corresponds to a segment that differs from its neighboring segments in texture. An edge in $G$ is assigned a weight representing the texture dissimilarity (Equation 5.1) between the corresponding pair of superpixels (Figure 5.5a). Because a tile-based image representation is used here, constructing $G$ requires iterating over all the tiles, connecting vertices within the same tile as well as vertices that lie on the boundary of adjacent tiles.

After constructing $G$, we use a modified version of the efficient graph-based segmentation (F&H) [3], originally formulated to work at the pixel level, to partition $G$. Starting from having each vertex being its own connected component, the algorithm examines all edges in a non-decreasing order with respect to edge weights. Given that the two end nodes of the edge under inspection belong to the connected components $C_1$ and $C_2$, these two connected components are merged if that edge has a weight less than or equal to a threshold $\min\{\text{diff}(C_1)+\tau(|C_1|), \text{diff}(C_2)+\tau(|C_2|)\}$. Here $\text{diff}(C)$ denotes the internal difference of $C$, calculated as the largest edge weight in the minimum spanning tree of $C$ (Figure 5.5b). Function $\tau(|C|) = \frac{k}{|C|}$, in which $k$ is a constant and $|C|$ denotes the number of vertices (superpixels) in the



(a) Building graph



(b) Merging two segments

Figure 5.5: (a) The dissimilarity of two adjacent superpixels (represented by edge weight) is calculated as the chi-square distance of two corresponding joint histograms. (b) The edge between the blue and red segments is eligible to merge the two segments if its weight $w_6$ is less than or equal to $\min(\max(w_1, w_2, w_3) + \tau(4), \max(w_4, w_5) + \tau(3))$.

connected component $C$; $\tau(|C|)$ decreases as connected component $C$ grows. Setting $k$ to a larger value leads to larger segments consisting of more superpixels.

## 5.5 Hierarchical Microstructures Exploration

The hierarchical segmentation procedure starts from the coarsest resolution and gradually moves to finer resolutions (i.e. from top to bottom level in the Gaussian pyramid). We first create a separate graph for each segment obtained in the previous (low resolution) level and then apply the aforementioned graph-based segmentation procedure to subdivide that segment. The hierarchical approach establishes the parent-child relationships between segments, which can be represented as a tree of segments across scales.

Because superpixels are generated independently before applying the hierarchical segmentation, the boundaries of segments at a previous (low resolution) level may not align perfectly with the superpixels in the following (high resolution) level. We address this inconsistency using a majority vote to assign superpixels in the child level to a parent segment.

After building the segment tree, each intermediate segmentation consists of the segments that correspond to the leaf nodes in the pruned segment tree. We use a hierarchical exploration strategy similar to Ip et al. [99] and build a tool that allows users to unfold a leaf node by clicking on the corresponding segment. The newly-added child nodes of that unfolded leaf node become the leaf nodes in the updated pruned tree, leading to the subdivision of the clicked segment. The colors

Table 5.1: The images used in the experiment. Sizes are measured in gigapixels.

| Image | Height | Width | Size | Levels |
|---|---|---|---|---|
| P1. Small fly | 5,632 | 11,776 | 0.06 | 5 |
| P2. Armadillidiidae | 18,176 | 8,960 | 0.16 | 6 |
| P3. Ant holding a fly | 12,544 | 18,944 | 0.23 | 6 |
| P4. Zebrafish embryo [5] | 49,152 | 122,880 | 6.04* | 8 |

* The original resolution is 281 gigapixels.

of segments reflect the changes made to the tree interactively.

## 5.6 Experiments

The proposed method was applied to four images. The largest image (P4) is the zebrafish embryo image downloaded with the Journal of Cell Biology (JCB) DataViewer[1]; the remaining three (P1–P3) were taken from the GigaPan website. Table 5.1 summarizes the size of the images and the number of pyramid levels. We downsampled P4 from the original resolution of 281 gigapixels to 6.04 gigapixels to reduce processing time.

P1–P3 were acquired with a scanning electron microscope (SEM); P4 was acquired with a transmission electron microscope (TEM). A typical SEM image shows the sample surface characteristics with a large depth of field, which provides a three-dimensional sense of the surface. On the other hand, TEM images capture the internal structure at high resolution by transmitting electron beams through a thin specimen. Here we report only the selected visualizations generated using P2 and P4.

[1]http://v.jcb-dataviewer.glencoesoftware.com/webclient/img_detail/201/

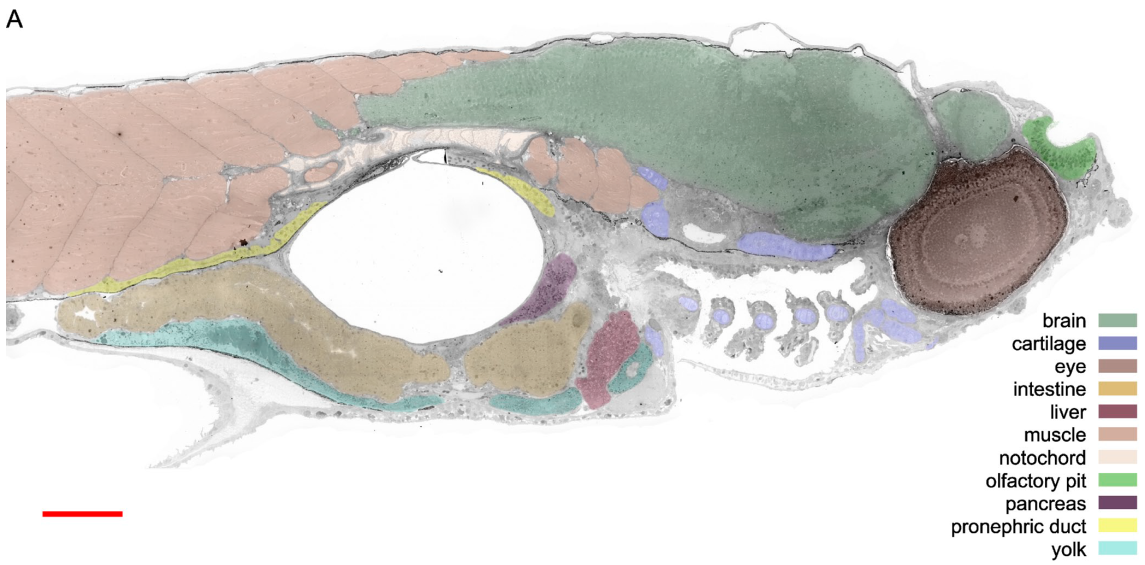


Figure 5.6: The 11 types of tissues in the zebrafish embryo (P4). Image courtesy of Faas et al. [5].

### 5.6.1 Dataset 1: Zebrafish

The embryo sample used in P4 consists of eleven types of tissues (Figure 5.6): brain, cartilage, eye, intestine, liver, muscle, notochord, olfactory pit, pancreas, pronephric duct, and yolk. We compare our results with tissue boundaries manually marked by domain experts [5] in the following.

Figure 5.7 shows an example of the hierarchical segmentation of P4. For simplicity, only part of the whole segment tree is shown. In the first row the root node, which represents the whole image, unfolds into node 5 and node 6 that correspond to segments 5 and 6 in the second row. Segment 5 further subdivides into child segments including segments 26 and 34 in the third row. In the segment tree, each node (except for the root) corresponds to a segment obtained through the graph-based segmentation. The tree edges between nodes 5 and 26 and nodes 5 and 34 represent the parent-child relationships between them.

The user interaction facilitates the reconfiguration of the segment tree following the steps from the first row to the third row in Figure 5.7. Initially, the segment tree contains only one (root) node, which corresponds to the yellow segment. Upon clicking the yellow segment, the root node unfolds into its child nodes (e.g. nodes 5 and 6). The two nodes correspond to two segments (blue and dark slate gray) in the second row; the dark slate gray segment contains eye tissues. Another click on the blue segment in the second row further subdivides the blue segment into a few smaller child segments (e.g. the red and brown segments in the third row). The click triggers the unfolding of node 5 into its children nodes 26 and 34 in the segment tree. If the users keep unfolding node 34 (and its child), they can eventually reach the visualization shown in the first column of Figure 5.8, which separates the muscle tissue from the intestine tissue.

Many tissues in the image, such as the muscle, intestine, and pronephric duct tissues (first column of Figure 5.8), are indistinguishable at low resolution. Users can hardly detect the subtle differences in texture because the high-frequency information is missing after zooming out. Our method, by enabling user intervention, provides corresponding color-based highlighting such that the muscle-intestine tissue boundaries are easily observable. The muscle tissues can also be split into several regions separated by a thin dark boundary (second column of Figure 5.8).

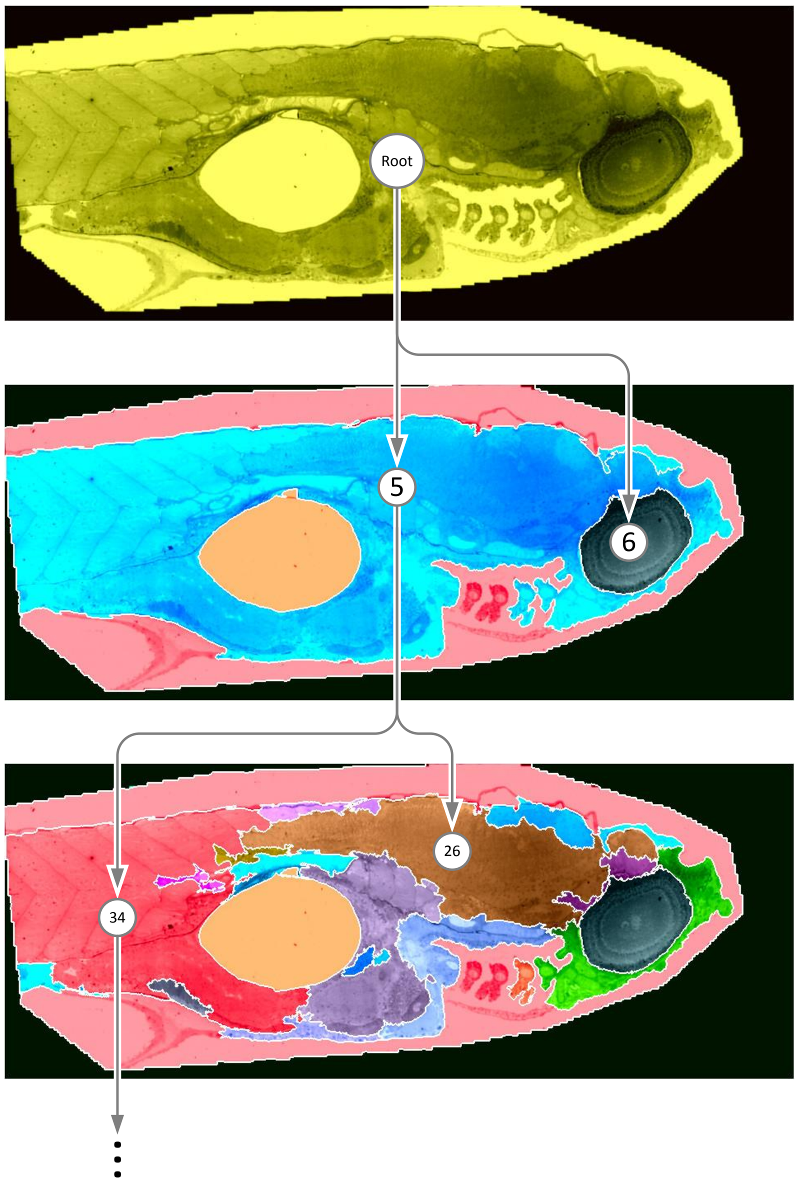


Figure 5.7: With our visualization tool, clicking the yellow segment in the first row subdivides the top-right region of the zebrafish embryo into segment 5 (blue) and segment 6 (dark slate gray). Another click on segment 5 subdivides segment 5 into segment 26 (brown) and segment 32 (red).

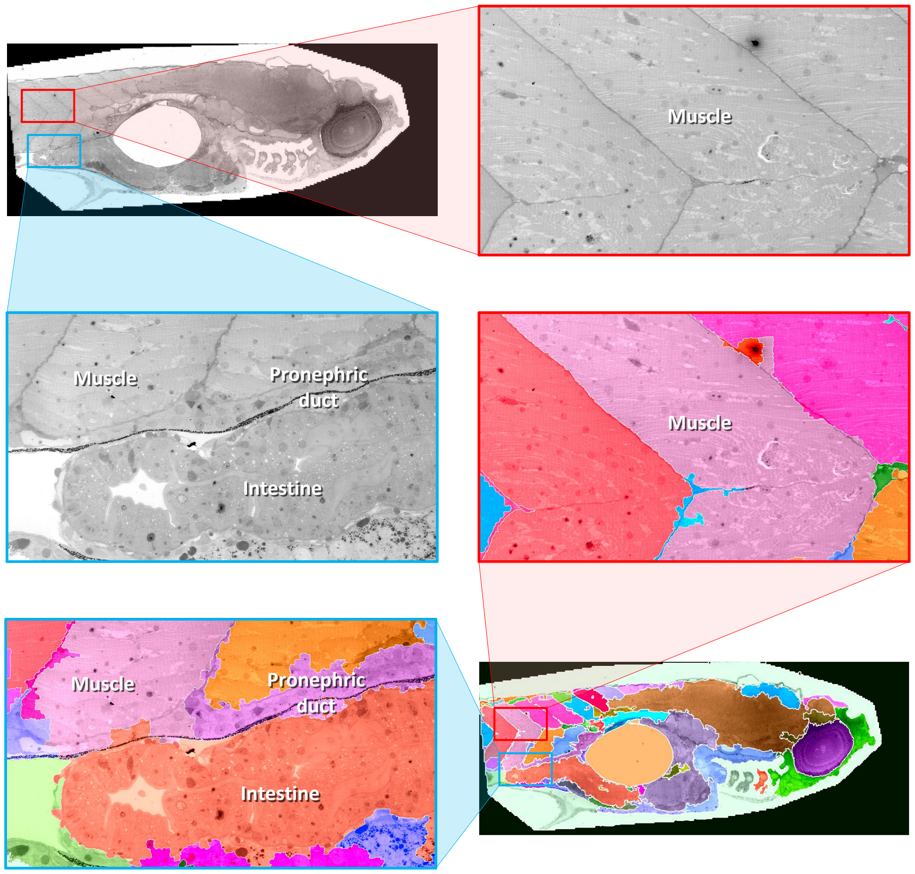


Figure 5.8: The texture differences in the down-sampled grayscale images are more distinguishable with color highlights. First column shows the boundary between the muscle, intestine, and pronephric duct tissues; second column shows the boundary between muscle tissues.

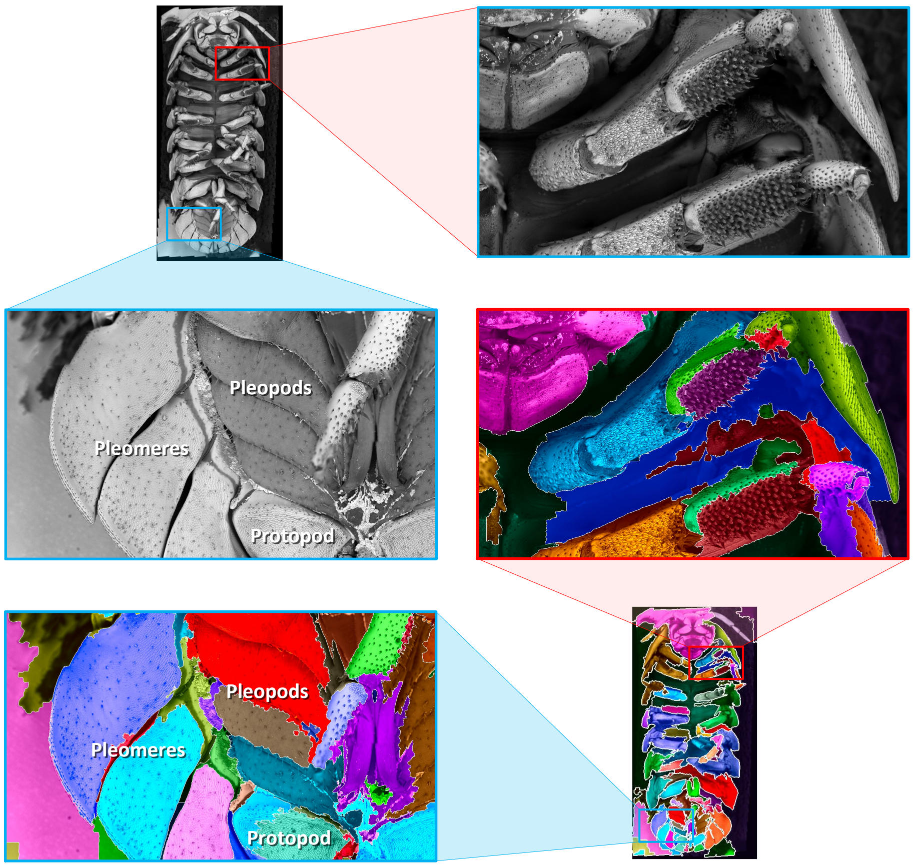


Figure 5.9: First column shows the separation of region with spikes in the leg from abdomen; second column shows the segmentation of protopod, pleomeres, and pleopods.

### 5.6.2 Dataset 2: Armadillidiidae

P2 is an image of an Armadillidiidae woodlouse. Part of the jointed leg has many dark spikes settled on the surface (first row of Figure 5.9). These spikes result in change of texture. Nevertheless, such change can be easily overlooked at low resolution because the abdomen part is also dark. The different coloring resulting from the proposed method hints the texture differences in the two adjacent regions. The second row of Figure 5.9 shows the segmentation result of protopod, pleomeres, and pleopods.

Table 5.2: The running time (m) of four most time-consuming processes and the total running time of the proposed method.

| Image | NRLBP | SLIC | Build graph | RAG/tree | Total |
|---|---|---|---|---|---|
| P1 | 0.08 | 0.22 | 0.65 | 0.12 | 1.10 |
| P2 | 0.21 | 0.52 | 1.13 | 0.29 | 1.55 |
| P3 | 0.25 | 0.74 | 1.58 | 0.42 | 3.05 |
| P4 | 8.99 | 16.22 | 50.94 | 18.70 | 96.79 |

### 5.6.3 Processing Time

Table 5.2 shows the total processing time for the hierarchical segmentation. For all images (P1–4), over half of the time was spent on constructing the graph, which involves creating the joint intensity-NRLBP histograms and calculating the edge weights ($\chi^2$ in Equation 5.1). The processing time reported here is based on an Intel Xeon 2.6GHz CPU. It has been shown that the calculation of LBP and $\chi^2$ can be substantially accelerated on a GPU [100]. We expect a similar degree of acceleration when we migrate the calculation from CPU to GPU.

### 5.6.4 Discussion

Lighting conditions influence segmentation results significantly because inconsistent lighting induces a shift in intensity values and therefore leads to considerably different intensity-NRLBP joint distributions. As a result, two regions of the same texture with different lighting conditions may end up being marked with distinct signatures. Such inconsistent lighting is especially apparent in SEM images that contain occlusions and shading. In addition, perspective distortion prevents precise texture description. Re-adjusting the lighting condition or using the earth mover's distance may improve the segmentation quality, thus leading to more accurate boundaries in the visualization.

A limitation of the hierarchical segmentation is that the segmentation of current level is based on the results of preceding levels. Errors in earlier stages therefore propagate to subsequent levels. A possible fix to this problem is including another bottom-up pass or interleaving top-down and bottom-up passes. Nevertheless, the computation load would increase dramatically with the number of passes. Also, we could start from some level in the middle of the pyramid if we know in advance the level range in which textures of interest are best separable. Limiting segmentation to those levels reduces the errors possibly introduced by segmenting in other irrelevant levels. However, doing this requires hints from users that may not be available in an exploratory context.

## 5.7 Conclusions and Future Work

The disparity between image size and screen resolution poses new challenges to the analysis of subtle microstructures. This work presents a method to assist exploration and analysis of gigapixel images by highlighting regions with different textures. We use a multiscale hierarchical image segmentation algorithm to identify regions that differ in texture based on the joint distribution of intensity and noise-resistant local binary pattern. These segments across resolutions are organized into a tree to facilitate the exploration of intermediate segmentation results interactively. The texture-based color highlights provide visual feedback to assist users in determining which parts of the image need more detail.

In this work, the texture descriptor is hand-crafted specifically for the targets in images. A possible future extension of this work is to identify robust texture and non-texture descriptors for evaluating similarity between the segments, so as to detect similar objects and structures. We will introduce the method we developed to derived useful features automatically using machine learning techniques in the next two sections.

# Chapter 6: Visualization of Live-cell Imagery

## 6.1 Introduction

Many studies of complex biological systems rely on live-cell imagery to measure, qualitatively and quantitatively, temporal changes in cell structure and behavior. These biologically-relevant changes, such as protein concentration, cell morphology, and geometry, are closely related to cell proliferation and cell life cycle. Several studies have shown that traumatic brain injury (TBI) causes progressive neurodegenerative changes and consequent dysfunction. However, the underlying mechanisms of such changes are not yet completely understood. Recent clinical studies on adult mice have linked TBIs to microglial activation, which was found to last for years after the initial brain trauma in human subjects as well [101]. Furthermore, it has been shown that, after TBI, the expression of several proteins increases significantly (e.g., cyclin G1, phosphorylated retinoblastoma, E2F1 and proliferating cell nuclear antigen, poly ADP-ribose polymerase) [102, 103]. The identification of highly expressed proteins upon TBI has enabled the definition of possible therapeutic strategies to reduce TBI-induced neuronal death [102]. In general, a better knowledge of neuronal death mechanisms, and in particular of the proteins involved, could lead to more effective therapeutics for TBI-induced neurodegenerative pathologies. In this work, we focus on identifying neuronal death with live-cell, time-lapse imagery.

Besides the challenges in recognizing complex objects in microscopy images, analyzing live-cell images is particularly difficult because active cells can move in all directions when the images are acquired. When a cell moves vertically across different focal planes, we need to adjust the focus of the microscope to capture crisp images of that cell. Furthermore, slight changes in the focus over time, a phenomenon referred to as focus drift, may degrade the image quality significantly. A common practice to accommodate for cell movements and focus drift is taking multiple images at various focal planes (or depths) at each time point; this practice ensures all cells are in-focus in at least some of the images. For a cell at a given time point, we need only a small number of in-focus images to determine the state of the cell and discard the other blurry, out-of-focus images.

In the past few years, convolutional neural networks (CNNs) have been successful in many biomedical applications such as microscopy image segmentation [104] and subcellular feature detection [105]. Most modern network architectures resemble that of AlexNet [60], which interleaves convolution layers and pooling layers to reduced spatial resolutions progressively. This arrangement allows the network to learn higher-level features in deeper (later) layers by combining lower-level features in shallower (earlier) layers.

In contrast to the work that applies 3D convolutions to address the temporal dimension of videos [106, 107], here we use 3D convolutions to combine features along the (third) dimension formed by the depths of various imaging focal planes. The

CNN therefore must learn to differentiate in-focus images from out-of-focus ones, and then selectively combine information useful for predicting cell states. Because the features of a layer are built upon those of the previous layer, the CNN-derived features should describe a cell from increasingly higher levels as we go deeper into the network. Ideally, we would want to use the (deep) features to identify potential intermediate states between the transition of cell states (from alive to dead). These intermediate states can then be used to create a graph-based visualization summarizing cell state transitions [55]. As a first step, we present here a user-mediated color assignment scheme based on deep features, and then use the assigned colors in our visualization.

We use 3D CNNs to determine the cell state independently at each time point (Section 6.3). Targeting exploratory data analysis, we build an interactive visualization tool that annotates cells by a user-defined transfer function, which maps abstract features representing cell characteristics to semantically-meaningful colors (Section 6.4). This color assignment scheme allows users to create informative visualizations despite their limited understanding of the underlying abstract features, such as the deep features derived automatically by CNNs and other sophisticated features. The tool also creates a visual summary to identify when and which cells change their states without inspecting each frame (Section 6.5). Although many studies have exploited the power of CNNs for computer vision applications, there have been hardly any studies that explore the potential of CNNs for visualization. Here, we use these deep features as a set of robust features that allow our color assignment scheme to create annotations depicting high-level concepts, which are not easily defined by hand-crafted features and conventional low-level features [99].

## 6.2 Related Work

### 6.2.1 Convolutional Neural Networks

Most ordinary CNN architectures more or less resemble the basic construct of AlexNet [60] but differ in depth. For example, GoogLeNet [108] has 22 trainable layers, much deeper than AlexNet, which has seven trainable layers. At this scale of depth, kernel sizes have to be kept small to maintain a manageable model complexity. This limitation in kernel size does not restrict the power of CNNs. In fact, deliberately stacking multiple convolutional layers with small kernels leads to a large receptive field, which is comparable to that of a large kernel but with fewer parameters [109]. Although research in complex problems tends to create deeper networks, simply stacking convolutional layers in a CNN does not always lead to better performance because of the increased model complexity. Addressing the deficiency of conventional CNNs, a 152-layer residual network [67] shows significant improvement over the state-of-the-art architectures.

Given a sequence of training examples, the training procedure starts with a forward pass through the network, calculates a loss value with respect to certain criteria, and then adjusts the weights associated with the neurons in the network accordingly by the backpropagation algorithm. Training a CNN appropriately can be challenging because of problems such as overfitting and premature convergence, which deteriorate performance significantly. Dropping connections between random neurons (i.e. removing them from the network temporarily) avoids co-adaptations among them [8], thereby reducing the chance of overfitting but also slowing down convergence. The slow convergence of CNNs is partly due to the frequent changes in the distribution of a layer's input during training. Batch normalization [6] addresses this problem by normalizing each feature independently to have a zero sum and unit variance distribution.

### 6.2.2 3D Convolutional Neural Networks

Although most studies involving CNNs have focused on images, recently CNNs have also been successfully applied to videos. Videos can be regarded as frame stacks along the temporal dimension, which can be treated as an additional spatial dimension [106] with 3D convolution (compared to 2D convolution for 2D images). Various ways of combining information across frames have been tested [110], which show a slightly better performance using 3D convolution. A CNN with 3D convolution is shown to outperform existing techniques, including other CNN-based approaches, on various recognition and classification tasks in large video collections [107]. Learning from entire videos instead of reduced video clips demonstrates substantial improvement over past methods [111]. Besides videos, the third dimension in 3D convolutions can also be used to address the spatial relationships for 3D data. Several groups have used 3D convolutions for segmentation and detection of target objects in volumetric electron microscopy images [112, 64, 113, 114]. Other groups have used 3D convolutions for 3D shape retrieval and completion [115, 116].

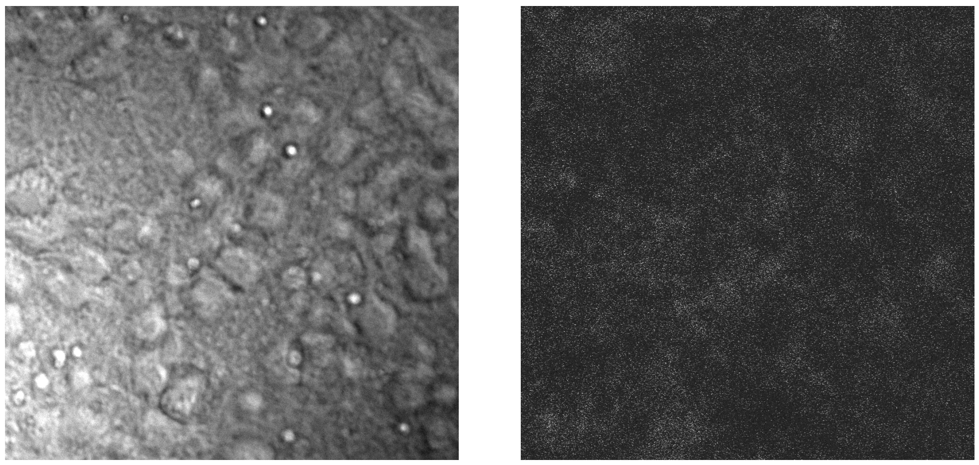
### 6.2.3 Data

We use a time-varying dataset that contains 86 microscopy images of adult mice neurons taken every 15 minutes. The live cells in the first two time points are manually labeled. Each image has two channels: the ordinary phase contrast channel and the fluorescence channel, which measures the mitochondrial potential. For each time point, ten images of size $1024 \times 1024$ are captured at different focal planes. The data at a time point $t$ can be regarded as a stack of ten images of different depths. In summary, the dataset is a 5D dataset of dimensions $(t, x, y, depth, channel) = (86, 1024, 1024, 10, 2)$.

For example, in Figure 6.1 we compare two images, one out-of-focus and one in-focus, captured at the same time point but at a different depth. Both channels in the out-of-focus image (Figure 6.1a) are blurry, obscuring the details in cell structures that are critical to determining cell states. On the other hand, the in-focus image (Figure 6.1b) provides much more information with respect to the detail interior structures of the cells and the distribution of mitochondrial potential.

## 6.3 CNN-based Feature Extraction

We use depth as the third dimension for the 3D convolution [106, 107] in addition the typical $x$ and $y$ spatial dimensions in our CNN. Similar to the way spatial information is combined in the $x$ and $y$ dimensions, the depth dimension of the kernels combines features derived from images taken at different focal plane planes. In the following, we describe the architecture of CNN for predicting the cell state of the


(a) $t = 1, depth = 4, channel = 1$(left); 2(right)

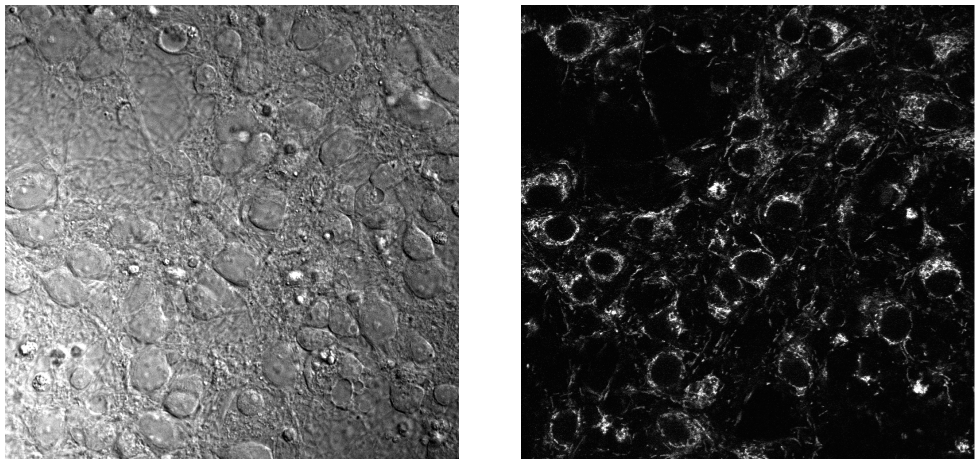

(b) $t = 1, depth = 7, channel = 1$(left); 2(right)

Figure 6.1: This example shows (a) an out of focus image and (b) a in focus image taken at the same time ($t = 1$) but at a different depth. The out of focus image provides much less useful information because the features are blurry.

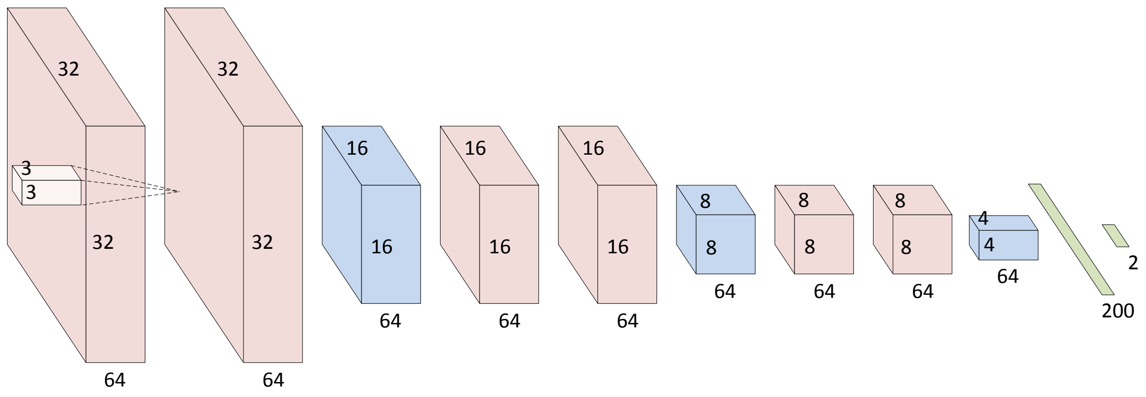


Figure 6.2: Our CNN has three groups of layers. Each group includes two convolutional layers (red), each with 64 kernels, followed by a max-pooling layer (blue) that reduces the spatial resolution (including depth) by half. For simplicity the drawing only shows the $x$ and $y$ spatial dimensions, omitting the depth dimension. The outputs of the first and second fully-connected layers (green) are 200 and 2 dimensional; they correspond to the high-level pixel representations for visualization and the probabilities of cell states.

center pixel that resides in a 4D input patch of dimensions $(x, y, depth, channel) = (65, 65, 10, 2)$. The input patches are sampled from the image stack at a given time point.

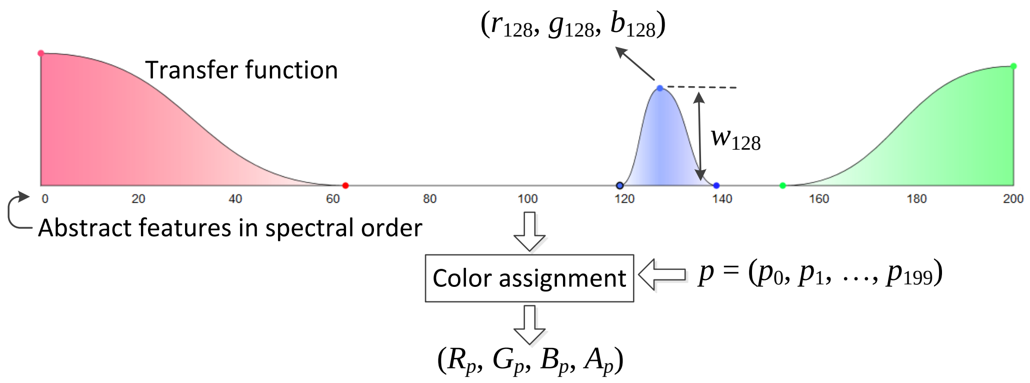
The CNN is composed of three groups of layers, each containing two $3 \times 3 \times 3$ convolutional layers with 64 kernels followed by a $2 \times 2 \times 2$ max-pooling layer that reduces the spatial resolution in all three dimensions (including depth) by half. The last pooling layer (of the third group) is followed by the first fully-connected layer with 200 neurons, which are then connected to the second fully-connected layer that outputs the prediction probability of cell states. In this work, we solve a binary classification problem (i.e. live cell versus dead cells and others), and therefore the number of outputs of the second fully-connected layer equals two. See Figure 6.2 for a graphical representation.

We apply batch normalization [6] to all convolutional layers and a dropout [8] of probability 0.5 for the first fully-connected layer. We use Parametric Rectified Linear Unit [7] as the activation function except for the second fully-connected layer, which uses sigmoid activation to generate output between 0 and 1. After training the CNN, our color assignment scheme uses the 200 outputs extracted from the first fully-connected layer as the pixel representation.

## 6.4 User-mediated Color Annotation

### 6.4.1 Color-based Annotation

The visual annotations used in our visualization tool distinguish cells of different states by colors, which provide salient visual hints to users. The tool tracks and establishes the correspondence of cells across different time points to allow coherent cell annotations over time. In time-lapse live-cell imaging, cell tracking and lineage construction is a complex problem that requires a global context for long-term interrelationship across various frames [117]. Because neurons in our target dataset do not undergo cell division, we can use an object tracker without lineage construction. Here we choose Kernelized Correlation Filter (KCF) [118] to track cell movements in adjacent frames. Individual KCF trackers, one for each cell, update the positions of the bounding boxes frame-by-frame and construct the complete paths of cell movement. The initial cell bounding boxes in the first frame are labeled manually. Expert biologists have visually verified the correctness of both the initial bounding boxes and the tracking results.



$(r_{128}, g_{128}, b_{128})$
Transfer function
$w_{128}$
0
20
40
60
80
100
120
140
160
180
200
Abstract features in spectral order
Color assignment
$p = (p_0, p_1, \ldots, p_{199})$
$(R_p, G_p, B_p, A_p)$

Figure 6.3: After extracting 200 deep features, we assign each pixel $p$ a color $(R_p, G_p, B_p, A_p)$ based on the user-defined transfer function, which can be easily modified after the 200 features are spectrally ordered along the $x$-axis.

### 6.4.2 User-mediated Color Assignment

We now describe a user-mediated color assignment scheme (Figure 6.3) to convert the 200 deep features, extracted from a trained CNN, into the colors used in our visualization. In the following we use $(p_0, p_1, \cdots, p_{199})$ to denote the $L_2$-normalized features extracted for a pixel $p$. Each deep feature $p_i$ is assigned a weight $w_i$, whose value is determined by the curve given the control points. Each control point is associated with a color assigned by users. After interpolation the curve therefore creates a transfer function that defines a mapping from $i$ to the corresponding color $(r_i, g_i, b_i)$ and weight $w_i$. The color assigned to the pixel $p$, denoted by $(R_p, G_p, B_p)$, is generated by a weighted average of $(r_i, g_i, b_i)$ using $p_i w_i$ as the weight (Equation 6.1, $G_p$ and $B_p$ are calculated similarly). The $i$-th feature will contribute more to the aggregated color of $p$ when $p_i$ and $w_i$ are larger. The opacity assigned to the pixel $p$, denoted by $A_p$, is the sum of the weights used in color aggregation. The opacity $A_p$ represents the importance of the pixel $p$ in comparison

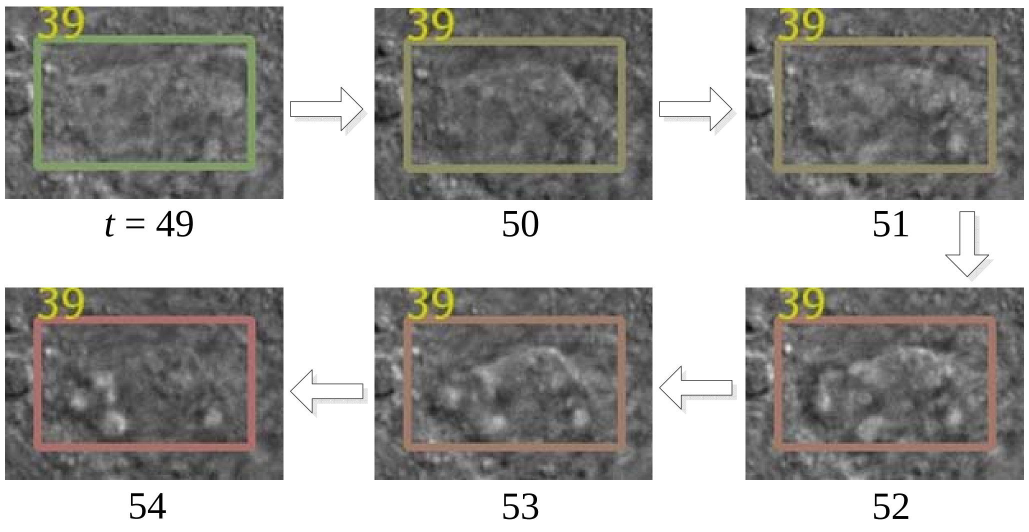


Figure 6.4: The change in color (from green to red) of cell 39 from frame 49 to frame 54 suggests that the cell is dying.

with other pixels.

$$R_p = \sum_{i=0}^{199} (p_i w_i) r_i \tag{6.1}$$

$$A_p = \sum_{i=0}^{199} p_i w_i \tag{6.2}$$

Finally, we calculate the color of a bounding box by blending the colors of the pixels inside of the bounding box with respect to their opacities. By augmenting the bounding boxes with the calculated composite colors, we can effectively annotate cells by their characteristics. For example, the transfer function in Figure 5.2 leads to the annotations in Figure 6.4, in which the change in color matches the change in the appearance of a cell that died between frames 49 and 54.

### 6.4.3 Similarity-based Feature Reordering

With the design widget, users can assign colors and weights to nearby features in groups based on their proximity along the $x$-axis. Nevertheless, the high-level features extracted from the CNN are 200-dimensional. Further, some, but not all, of the extracted features could be closely related to each other. If two unrelated features (e.g. $i$- and $j$-th feature) are in close proximity along the $x$-axis of the design widget (Figure 6.3), assigning different colors (e.g. $(r_i, g_i, b_i)$ and $(r_j, g_j, b_j)$) and weights (e.g. $w_i$ and $w_j$) to them may be difficult. In the worst case, such overhead could even overshadow the benefits of applying the deep-learning-assisted approach.

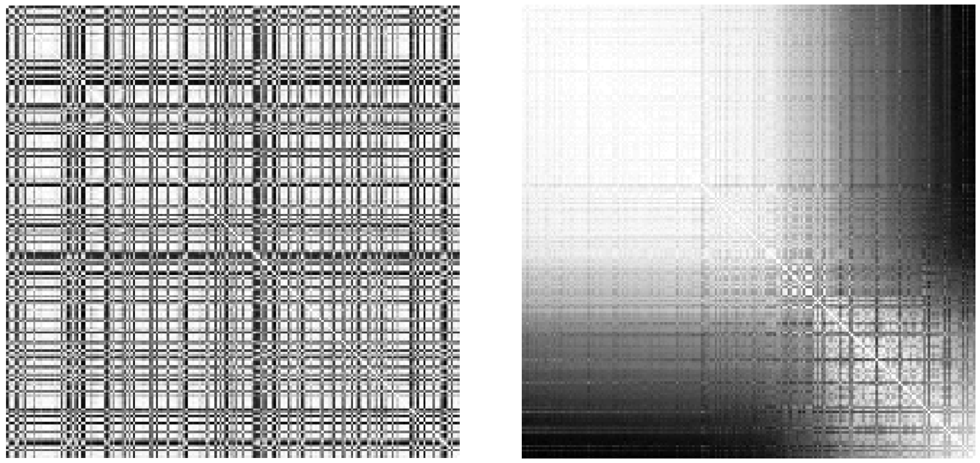
A proper ordering of features will therefore reduce the complexity of transfer functions; for example, users can define the same transfer function using far less control points. We address this usability issue by spectral ordering [119] features along the $x$-axis in Figure 6.3 based on pairwise feature-feature similarity such that similar (dissimilar) features are near (away) from each other. For example, dissimilar features located in the left and the right end of $x$-axis can be assign distinct colors separately (i.e. red and green) in Figure 6.3. The similarity-based ordering therefore accelerates the design of transfer functions.

The spectral ordering of features consists of the following steps. First, We start with a feature-by-pixel matrix $M$ and compute a similarity (adjacency) matrix $A$ by multiplying the matrix by its transpose, resulting in a feature-to-feature matrix: $A = MM^\top$. Second, we calculate a diagonal matrix $D$, in which the diagonal elements $D(i,i)$ is the sum of the $i$-th row vector of $A$. The Laplacian matrix $L$ is then calculated as $L = D - A$. Based on the Laplacian matrix $L$, we calculate the eigenvector associated with the second smallest non-negative eigenvalue (the Fiedler vector). After calculating the Fiedler vector, we sort the original ordering of features in the input matrix $M$ using the values in the Fiedler vector. The result is an ordering of features where neighboring features are similar. The Fiedler vector and other eigenvectors associated with small eigenvalues also form the basis of spectral clustering [39].

We compare the similarity matrix of the 200 features before and after spectral ordering in Figure 6.5. With an arbitrary order, the features are disorganized as the bright entries, indicating similar feature pairs, scatter all over the matrix (Figure 6.5a). After spectral ordering, the bright entries are mostly near the top-left and bottom-right corners (Figure 6.5b). Similar features therefore locate in the left and the right along the $x$-axis (or the top and the bottom along the $y$-axis) of the similarity matrix.

### 6.4.4 Vector Quantization

The color assignment scheme requires all 200 features to update the assigned color when the transfer function changes. This requirement imposes a huge memory burden. For example, we would need 69 gigabytes of memory to load the features for all the 86 time points in our dataset. We could assign colors offline but that would compromise interactivity, which we consider crucial in the visual reasoning



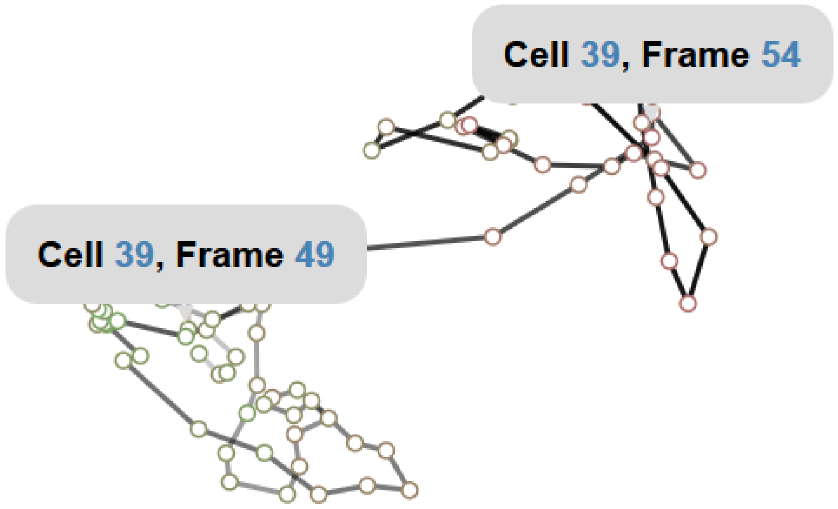
(a) Arbitrary order (b) Spectral order

Figure 6.5: This example shows the $200 \times 200$ feature-to-feature similarity matrix, in which bright entries indicate similar feature pairs, created using (a) arbitrary order and (b) spectral order. Similar features are closer to each other after organizing features in the spectral order, thus resulting in large bright regions in the similarity matrix.

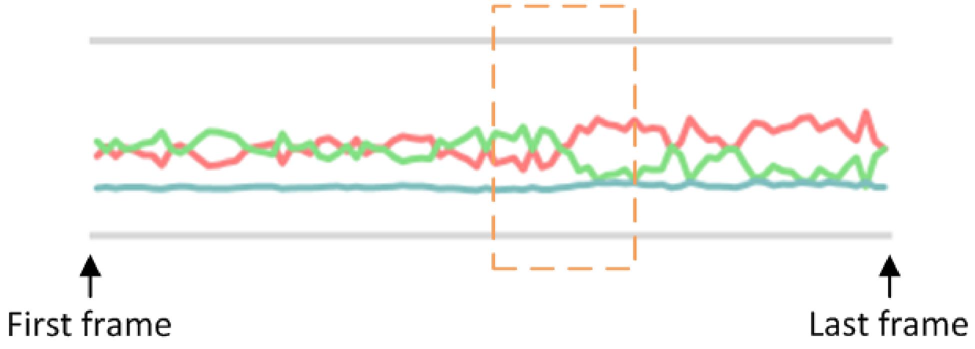
process. As a result, we apply vector quantization using an incremental $k$-means algorithm [120] to create a codebook with $k$ codewords (the centroids of clusters). After quantization, we need only a single index into the codebook to represent a pixel instead of the original 200 features. Throughout this work, we use $k = 256$ such that the indices of codewords are one byte in size.

## 6.5 Cell-state Trajectory Visualization

The visualization we have presented in Figure 6.4 is still a frame-by-frame playback of cells between two time points. Inspecting all frames visually may become tedious and ineffective as the number of frames grows. Here we assume that cell positions are unrelated to their states and use the position in screen space to represent the trajectory of change in cell states, leading to a visualization that shows a summarization over a period of time (Figure 6.6).



(a) Trajectory of a cell



(b) Color change over time

Figure 6.6: (a) The state change of a cell is visualized as a segmented line. The change in color from green to red indicates a state change of cell 39 from live to dead between frame 49 and 54. (b) The change in color of cell 39 can also be visualized separately with three lines, representing the RGB values over time. Here we can see the state change happens inside of the orange box.

The 2D layout of cells is determined as follows. We use the color histogram of pixels enclosed in the bounding box as cell representation at a specific time point. The dissimilarity between a pair of cells is assessed using the Earth mover's distance (EMD) of the corresponding pair of color histograms. We use the average EMD of three pairs of histograms (i.e. R, G, and B) as the final dissimilarity between pairs of cells. Given the pairwise dissimilarity matrix, we can now create a 2D layout of cells such that similar (or dissimilar) cells are closer (or farther) through multidimensional scaling [98]. After obtaining the 2D coordinates of cells, the trajectory of a cell is represented as a segmented line formed by segments connecting the cells in adjacent frames (Figure 6.6a). The grayscale levels (white to black) of segments represent the progress of time. For example, Figure 6.6a shows the trajectory of cell 39, which starts dying around frame 49 (c.f. Figure 6.4). Between frame 49 and 54 the location of cell 39 in the 2D layout changes from the bottom left to the top right while its color changes from green to red. Based on this visual representation, we build an interactive visualization tool that allows users to select cells with similar trajectory profiles. The tool also shows the change in color over time (Figure 6.6b). From left to right, color lines depict the changes in red, green, and blue over time for cell 39. During the time period within the orange box, the rising red line and the falling green line indicate the death of that cell.

## 6.6 Results and Discussion

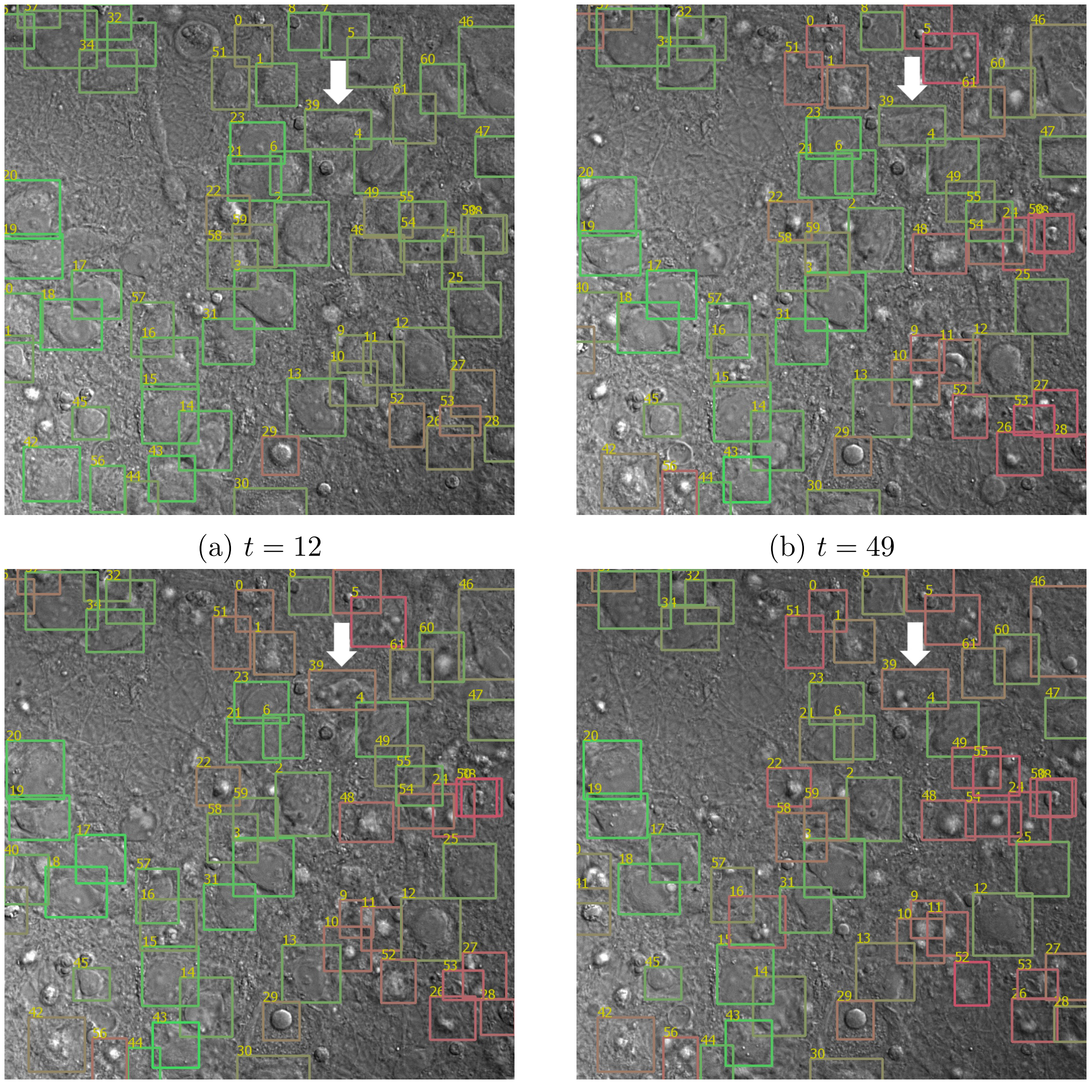
We implemented our CNN using the Caffe framework [121]. We used a total of 400K samples, divided equally between the two classes (i.e. live cells and others), to train the CNN. We used a stochastic gradient descent solver with 0.9 momentum and 0.0005 decay to minimize binary cross-entropy loss. The training procedure ends after 240K iterations using a batch size of 50.

### 6.6.1 Results

Most color-based annotations are consistent with the changes in cell states. Figure 6.7a–6.7d show the augmented frames at $t = 12$, 49, 54, and 78, respectively. Users can quickly identify which cells are dead by the color of their bounding boxes. Figure 6.8 shows the trajectories of cells 3, 38, and 39. Because the color assigned to cell 3 is always green, users can easily conclude that cell 3 stays alive from the first time point to the last. Both cells 38 and 39 died at some point but their trajectory profiles differ significantly. By looking at the color lines (top of Figure 6.8), we observe that the green line of cell 38 stays low and never grows back again once it starts decreasing. The green line of cell 39, however, shows a jagged pattern. After having a closer look in Figure 6.4 and Figure 6.7, we believe the fix-sized bounding boxes we used caused the jagged pattern because the bounding box no longer tightly fits cell 39 after it died. As a result, the bounding box inevitably includes many pixels that belong to the background, thereby introducing noise into our color assignment scheme. A dynamically sized bounding box or, even better, cell segmentation would alleviate this problem.

### 6.6.2 Discussion

Extracting features from our CNN requires a significant amount of time, about one hour per frame using an NVIDIA GTX 970 GPU. Instead of using the current CNN architecture that outputs features pixel-by-pixel, we can use the concept of fully convolutional network [66] to output features of nearby pixels together in one


(a) $t = 12$

(b) $t = 49$

(c) $t = 54$

(d) $t = 78$

Figure 6.7: This example shows the color annotations in four time points (a) 12, (b) 49, (c) 54, and (d) 78. Most cells are alive (green) at time point 12. They die (red) gradually in subsequent time points (i.e. 49, 54, and 78). Cell 39 (white arrow), which changes color from green to red, is dead sometime during frame 49 and 54.

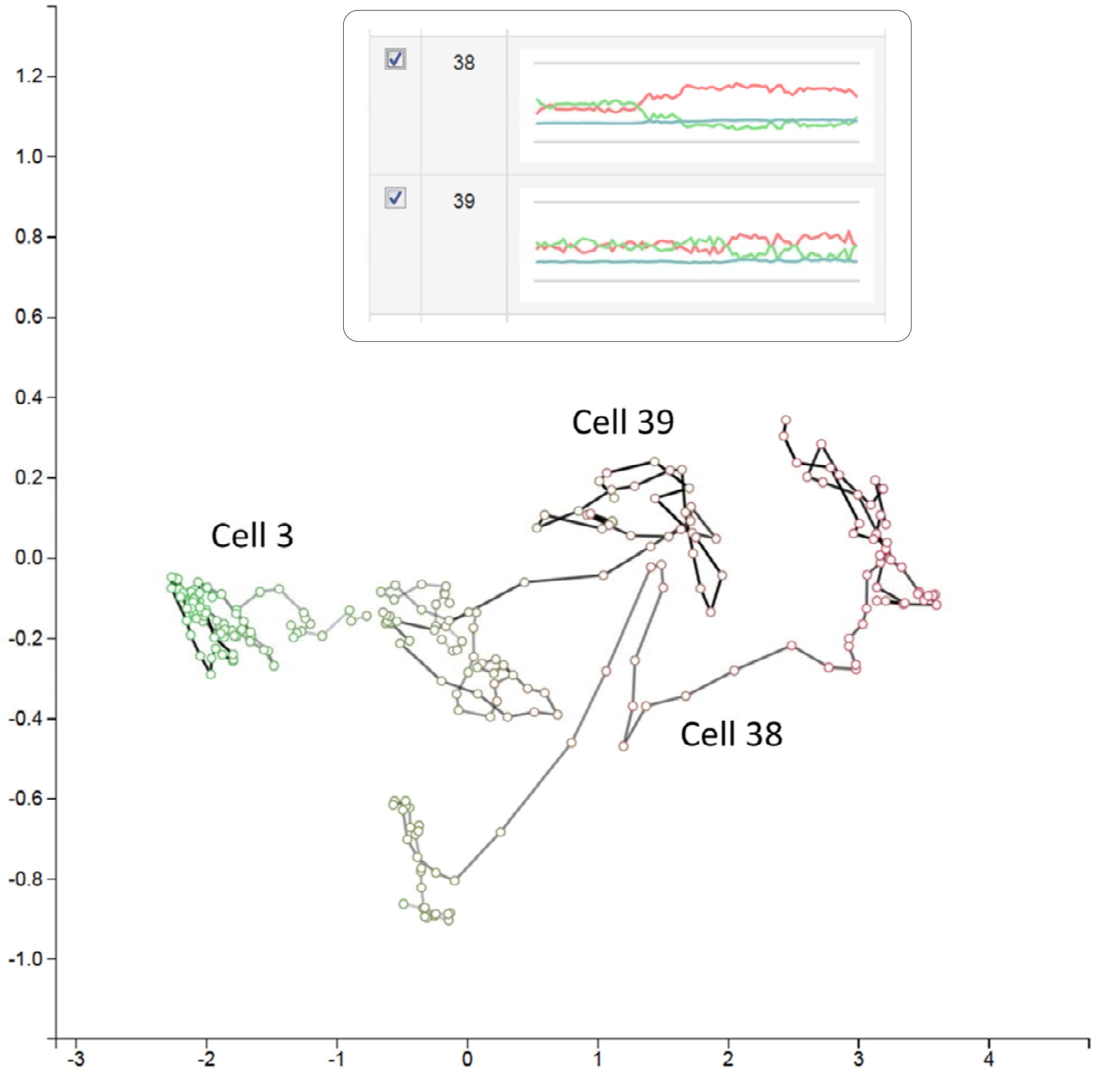


Figure 6.8: Our visualization allows users to select and compare cell trajectories without consulting different frames. This example shows three cells with distinct trajectories. Cell 3 stays alive whereas cells 38 and 39 died at specific frames, hinted by the drop in color green (and the growth in color red).

forward pass, avoiding redundant computation.

Though we tested the summarization-based visualization using only one time-varying dataset, we can repeat the same procedure for multiple datasets and create summary of the whole collection. This summary has many potential applications, such as analyzing the changes of cell behavior under different treatments.

In general, the high-level representations we extracted using the deep-learning approach can also serve as input to design other application-dependent visualizations for live-cell imagery. As microscopy data grows in size, appropriate visual representations that can provide a high-level summary are becoming ever more important to the analysis of complex data. The techniques and visualizations introduced here can facilitate the understanding of large complex time-varying data and lead to new findings and insights.

## 6.7 Conclusions and Future Work

We present a method that uses 3D CNN to detect and depict temporal changes in cell states in live-cell images. Based on the abstract features derived directly from the data, we build a visual analytics tool that allows users to create comprehensive color-based annotations of cells. Our tool also creates a summarization-based visualization that shows the change of a cell over time as a segmented line. This static 2D layout allows users to compare cells or search for cells with similar trajectory profiles. The promising results show that our tool can improve the understanding of complex time-varying datasets.

We plan to work with biomedical experts and apply the same analyses to other live-cell image datasets. We also plan to extend our deep-learning-assisted techniques for multi-dimensional volume visualization applications [72, 73]. We will show our preliminary result on volume rendering in the next section.

# Chapter 7: Deep-learning-assisted Volume Visualization

## 7.1 Introduction

Volume visualization typically involves three steps. First, users define the criteria (e.g. intensity and texture) that distinguish the structures of interest (e.g. soft tissue and bone). Second, based on these user-defined criteria, users choose an appropriate set of per-voxel features that span the low-dimensional subspace in which one can look for desirable solutions (i.e. visualizations). Finally, users interact with the visualization tool to create and modify the solutions. In the past, handcrafted features that correspond to specific user-defined criteria (e.g. size [122, 123], texture [124], and visibility [125]) have been successfully used as volumetric features for the second step. Nevertheless, as the complexity of the user-desired criteria grows, finding features that precisely describe the characteristics of the target structures becomes increasingly challenging. Conventional handcrafted features no longer suffice because they are defined locally without addressing the relationships among voxels in the global context, often crucial to characterizing complex structures.

Although the high structural complexity significantly hinders the manual search for suitable feature spaces in the second step of the current visualization workflow, users can still provide valuable domain knowledge in a different way. From a machine learning point of view, the criteria that distinguish different structures can be implicitly defined by a large number of examples that include as many structural variants as possible. In this paper, we present our approach in which we train a CNN as if we were solving a classification problem based on a given set of examples. After training, the CNN automatically derives a high-level data representation, thus creating a feasible feature space for the visualization of complex structures. Despite improvements in user interfaces [126, 127, 128] and semi-automatic techniques [129, 99], current methods proceed with the assumption that a suitable feature space is explicitly defined, not automatically learned. In contrast, the proposed deep-learning-assisted approach automatically creates a high-dimensional feature space in a data-driven way without an explicit specification from the user (Section 7.4).

In the derived feature space, voxels with similar characteristics are in close proximity. These similar voxels can therefore be described and selected for visualization with a representative characteristic feature vector, which essentially defines a point in the high-dimensional space. We can therefore extracting surfaces from voxels defined in a high-dimensional feature space (Section 7.5). We first convert the feature space to a binary scalar field and then extract a triangular mesh from that binary scalar field using the conventional marching cubes algorithm [130]. Similar to the conventional method, the binary scalar field is obtained by thresholding an intermediate scalar field, defined by the user through the design widget. This conversion can be done efficiently on a GPU with little computational overhead.

However, the characteristic feature vector becomes complicated to modify as its dimensionality grows. Previous work alleviates this problem by designing visualizations in a reduced space with manageable complexity [127, 131]. In this work, we present two techniques for this design problem. The first one reorders the features with respect to their similarity (Section 7.6). With this improved, similarity-aware feature layout, the conventional design widget can select groups of similar voxels in the original space without dimensionality reduction. This is complementary to the dimension-reduction-based techniques. The second one is a semi-automatic technique that generates a hierarchy of volume visualizations for users to choose from, thus providing another layer of abstraction on top of the high-dimensional feature space (Section 7.6.3).

Although modern rendering techniques and hardware can now render volumetric data interactively, we still need a suitable feature space that facilitates natural differentiation of target structures and an intuitive and interactive way for users to design visualizations. This paper presents our initial explorations in using deep learning methods to assist the design of volume visualizations. We expect that the techniques presented in this work will be useful in many visual computing tools for designing informative visualizations.

## 7.2 Related Work

### 7.2.1 Volume Visualization

Typical volume rendering techniques can be grouped into two categories: direct and indirect volume rendering. Direct volume rendering (DVR) techniques, such as volume ray-casting [132], compose a result image by aggregating the colors and opacities of relevant voxels calculated using a user-defined transfer function. Indirect volume rendering (IVR) techniques first generate geometric primitives as an intermediate representation and then render those primitives using conventional 3D computer graphics. For example, the marching cubes algorithm [130] extracts a triangular mesh that represents an isosurface in a structured grid.

Despite different ways of composing result images, DVR and IVR techniques all require a feature space suitable for the visualization criteria. Simple structures that are distinguishable by intensity values can be extracted and visualized as corresponding isosurfaces [130] following the IVR framework. In many applications, however, the complex structures and their surroundings are indistinguishable based on a single isovalue; these structures may be better extracted as subvolumes composed of voxels with intensity values contained in an interval [133, 134].

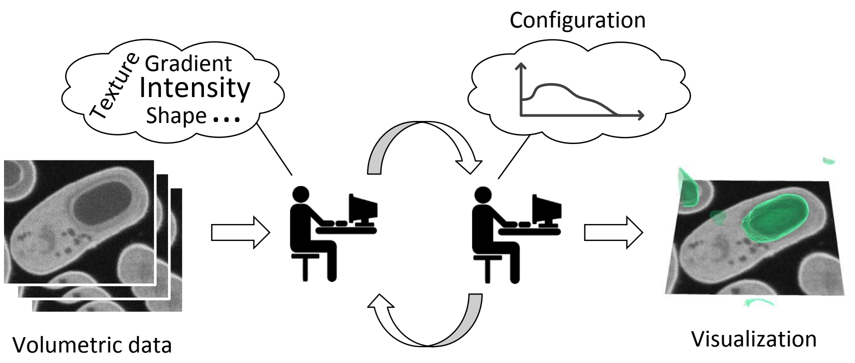
Whereas typical IVR techniques extract geometric primitives from a scalar field formed by intensity values, DVR techniques rely on a transfer function, sometimes defined in an alternative feature space derived from the intensity, to assign distinct colors and opacities to different structures. Common features such as gradient magnitude [126] and curvature [135] allow distinguishing regions with significant local changes in intensity. Other specialized features are used for structures with specific characteristics. For example, twenty texture features have been used to identify voxels with texture differences [124]. Evaluating the size of structures in scale-space at each voxel creates a scale field that can be used to highlight structures of different sizes [122]. An alternative way to assess structure sizes is by searching in all directions for neighboring voxels with similar intensities [123].

Whereas most volume visualizations are done manually, semi-automatic techniques automate specific time-consuming procedures to accelerate the design. Candidate isosurfaces that correspond to meaningful structure boundaries can be selected automatically based on gradient magnitudes [136]. Exploiting the proximity of voxels in the feature space, many semi-automatic techniques use clustering algorithms to group similar voxels, thus reducing the complexity of visualization design [129, 99].

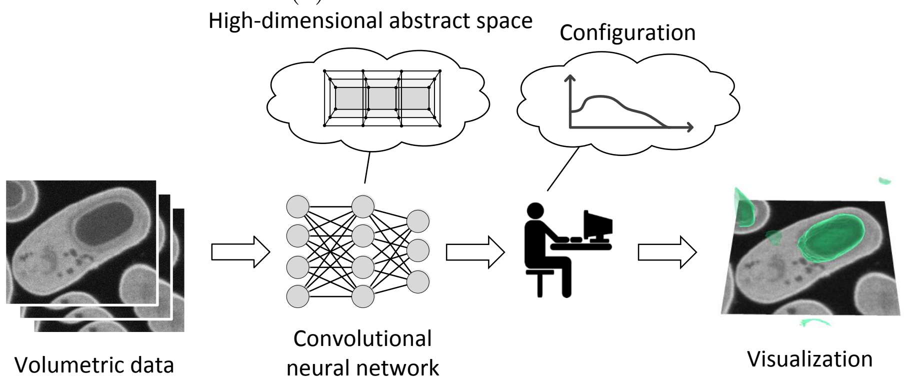
A prerequisite of successful volume visualizations, including those created by semi-automatic techniques, is that the selected feature space must differentiate the target structures. Finding such a feature space is therefore critical for volume visualization design. Nevertheless, existing feature spaces, including those formed by texture features, are too local to reliably differentiate complex structures when given only limited context around each voxel. In this work, we address this limitation by forming a feature space based on high-level features extracted automatically from a trained CNN. Based on the derived feature space, we then create visualizations following the IVR framework.

## 7.3 Motivation

Visualization is crucial to the analysis of volumetric data yet its design remains labor-intensive. Conventional design workflow assumes a feasible feature space with which users, after some trial and error, will eventually find configurations suitable for the target structures. Nevertheless, so far the feature space is always hand-



(a) Conventional workflow



(b) Proposed workflow

Figure 7.1: (a) Conventional workflow requires users to adjust both the feature space and the configuration. (b) Our deep-learning-assisted approach derives from labeled examples a feasible feature space automatically, effectively removing the need for an user-defined feature space.

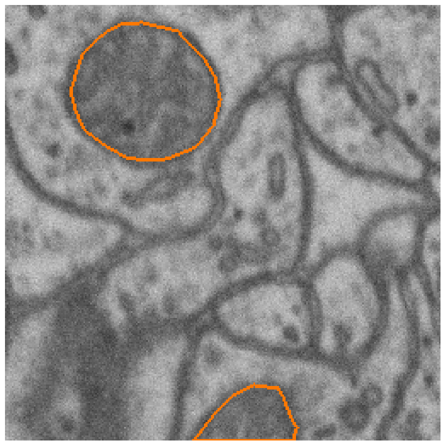
crafted with respect to specific application-dependent criteria. This manual procedure limits the feature space to incorporate only local features because of the limited capability of human to comprehend and integrate non-local properties defined in high-dimensional spaces. As structural complexity grows, the assumption of having a feasible feature space is increasingly unrealistic. When the visualization result does not suffice, users are faced with the non-trivial choice to modify the feature space, the configuration, or both in the worst case (Figure 7.1a).

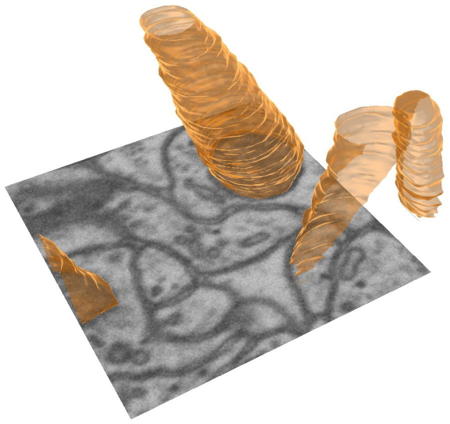
This observation motivates our deep-learning-assisted approach that learns a feasible feature space automatically when provided enough examples of the target structures. This approach adopts a workflow that effectively replaces the manual search of a feasible feature space by the concrete labeling of samples followed by deep learning (Figure 7.1b). In contrast to crafting features in a high-dimensional abstract space, the labeling task is much easier for typical users, who are domain experts familiar with the data but have limited knowledge about computational techniques.

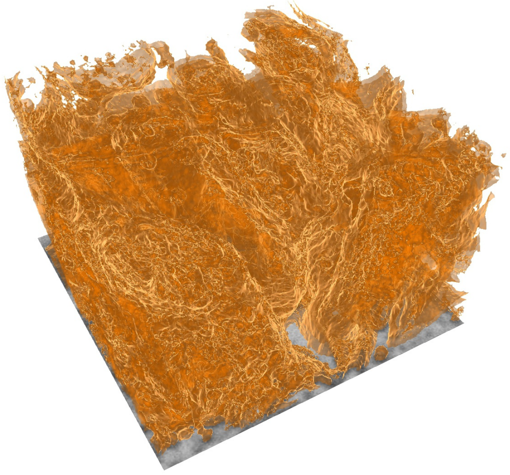
### 7.3.1 Conventional Volume Visualization Design

With imaging modalities such as X-Ray and computed tomography, structures of different intrinsic material densities (e.g. soft tissue and bone) are distinguishable by their distinct intensity values. This characteristic allows conventional intensity-based feature spaces to separate these structures effectively. For example, the boundaries of bone may correspond well to a certain isosurface. With other modalities, such as electron microscopy, the type of structures to which a voxel belongs is seldom decided solely by the voxel's intensity value without consulting the arrangement of the neighboring voxels. Large intra-class variations further complicate the decision. In addition, gradient only describes local changes in intensity values and lacks the capability of depicting the boundary of complex structures. As a result, intensity and gradient do not provide enough information for designing informative visualizations for complex structures.

For example in Figure 7.2 we compare the meshes formed by the ground truth labels of mitochondria (Figure 7.2b), generated using the conventional marching


(a) Raw image


(b) Ground truth


(c) Isosurface (isovalue = 112)

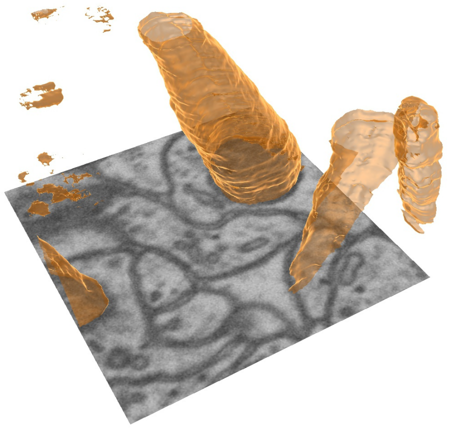
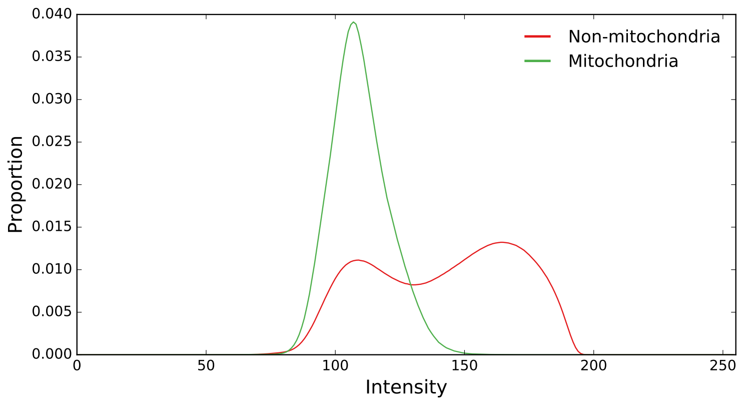

(d) Our visualization

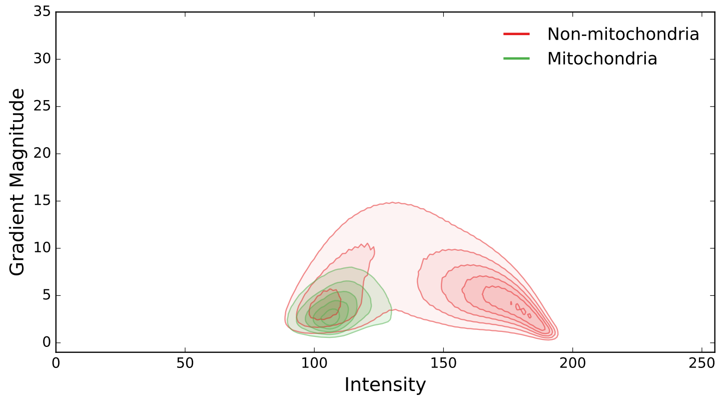
Figure 7.2: (a) In the hippocampus dataset, we find it challenging to differentiate mitochondrial regions (orange) from non-mitochondrial regions using conventional intensity-based feature spaces. (b) The ground truth mitochondria. (c) The isosurface of isovalue 112, which is the average intensity value of mitochondria, does not correspond well to the boundary of mitochondria. In fact, no single isovalue would be suitable for differentiating the mitochondria (c.f. Figure 7.3a). (d) Our visualization result is comparable to the ground truth because it is based on a feasible feature space derived automatically.

cubes algorithm (Figure 7.2c), and generated using our deep-learning-assisted approach (Figure 7.2d) for the hippocampus dataset (c.f. Section 7.7.3). Although the mitochondria are typically low in intensity, many non-mitochondria voxels, for example those belonging to the membranes, are also low in intensity (Figure 7.2a). As a result the isosurface of isovalue 112, which is close to the average intensity value of mitochondria, does not map precisely to boundaries of mitochondria, thus creating an incomprehensible visualization with severe occlusions of various structures. In contrast, our derived feature space leads to a visualization of mitochondria comparable to the ground truth labels.

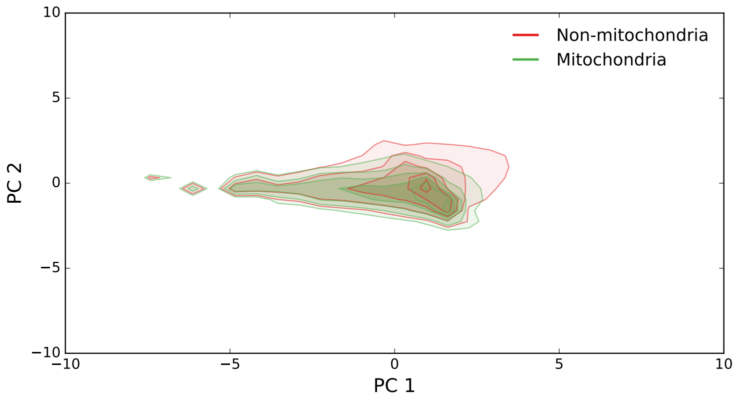
The previous example shows the deficiency of conventional feature spaces in depicting complex structures. In fact, a significant overlap of the two types of voxels (i.e. mitochondria and non-mitochondria) in the intensity histogram confirms that we cannot obtain clear boundaries using a single isovalue (Figure 7.3a). Adding gradient magnitude as a second dimension does not improve separation in the density plot either (Figure 7.3b). Besides intensity and gradient magnitude, Haralick features [77], a set of texture features calculated from a small local neighborhood, also lack the power of depicting mitochondria although they have been successful for other structures in computed tomography and magnetic resonance images [124]. Here we calculate seven Haralick features (i.e. energy, inertia, inverse difference moment, entropy, correlation, contrast, and sum of entropy) with the same configuration used in [124]. After projecting the voxels onto 2D using principal component analysis, the large overlapping near the center shows that the selected Haralick features do not separate the mitochondria from the rest of the image (Figure 7.3c).



(a) Intensity histogram



(b) Density plot of intensity and gradient magnitude



(c) Density plot of seven Haralick features after PCA

Figure 7.3: (a) The overlap in the intensity histogram shows that we cannot extract the mitochondria precisely using a single isovalue. The density plots, in which higher saturation indicates higher density, show that (b) intensity and gradient, and (c) seven selected Haralick features do not help extract the mitochondria either. The seven Haralick features are projected onto the first two principal components, “PC 1” and “PC 2”, obtained using principal component analysis.

### 7.3.2 Deep-learning-assisted Visualization Design

Previous studies on the hippocampus dataset have shown that features depicting both the complex shape of the mitochondria and the contextual information around voxels perform better in segmentation [137]. If we were to apply the conventional design workflow to create visualizations that show mitochondria and other complex structures presented in this paper, users would need to define such sophisticated features themselves. Although there may exist a combination of novel and established features properly configured for those specific targets, searching manually for that combination is time-consuming and challenging. On the contrary, our proposed approach uses a CNN to perform that search effectively and automatically. Users can therefore focus on creating useful volume visualizations based on the derived feature space.

Our deep-learning-assisted approach extracts high-level features (i.e. deep features) from a trained CNN (top of Figure 7.4). We first train the proposed CNN as if we were solving a voxel-wise classification problem. Because CNNs are powerful in finding suitable application-dependent features, we use those features to distinguish voxels that belong to various complex structures during rendering. The extracted deep features correspond to much higher level concepts when compared with local features such as gradient magnitude or textures, thus improving their capability in discerning complex structures.

Based on the derived feature space, we address practical issues when creating volume visualizations using deep features (bottom of Figure 7.4). Even for moderate-

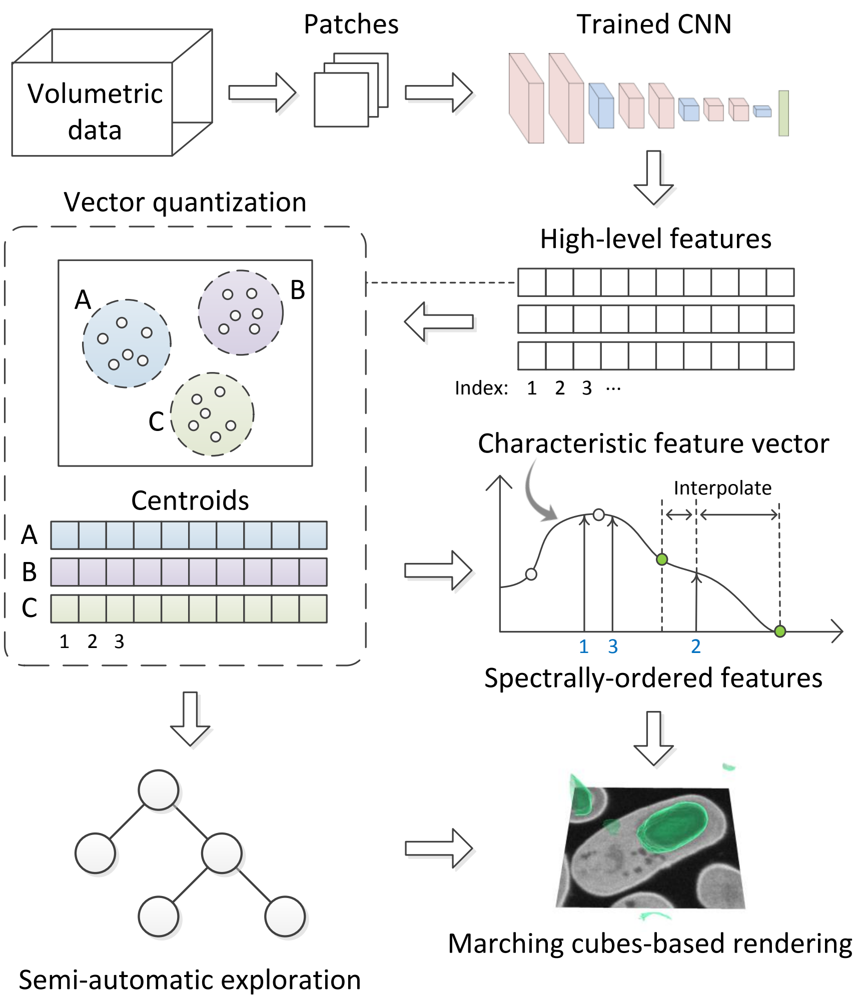


Figure 7.4: Our proposed deep-learning-assisted approach first extracts high-level features from a trained CNN. We then use vector quantization to encode the extracted high-dimensional features of each voxel by the nearest centroid found using $k$-means clustering. Users modify the visualization result, which is generated by a marching cubes-based rendering, either by editing the characteristic feature vector of a target subvolume or exploring a pre-generated subvolume hierarchy semi-automatically.

sized volumetric data, the high dimensionality of features could be overwhelming because of the limited memory available to a GPU. Therefore, we use vector quantization to compress the features to a manageable size. Furthermore, because the features that form the high-dimensional feature space are not independent, we apply a spectral method to reorder them. After feature reordering, users can easily change the visualization by modifying a characteristic feature vector through a simple design widget. We also use a semi-automatic method to accelerate the design of volume visualization by pre-generating a tree of visualizations, with which users can explore the volumetric data hierarchically and interactively.

## 7.4 CNN-based Feature Extraction

The high-level features are extracted from a CNN, which contains three groups of convolutional layers (Table 7.1). The network is similar to the previous one we used in Section 6 except that here we use 2D convolutions and the first convolution has a stride of two. Because of the stride-two convolution, this CNN reduces resolution more aggressively than the previous one. In each group, we stack two convolutional layers with $3 \times 3$ kernels to resemble the reception field of a $5 \times 5$ kernel but with fewer trainable parameters. We perform batch normalization for all the convolutional layers. Also, we use dropout [8] with a probability of 0.5 for the first fully-connected layer. The first fully-connected layer contains 200 neurons, which connect to neurons in the second fully-connected layer. Each neuron in the second fully-connected layer corresponds to an individual class in the data. We chose the

Table 7.1: The architecture of the CNN

| Layer | #channels | #neurons | Kernel | stride |
|---|---|---|---|---|
| Convolutional | 64 | $32 \times 32$ | $3 \times 3$ | 2 |
| Convolutional | 64 | $32 \times 32$ | $3 \times 3$ | 1 |
| Max-pooling | 64 | $16 \times 16$ | $2 \times 2$ | 2 |
| Convolutional | 64 | $16 \times 16$ | $3 \times 3$ | 1 |
| Convolutional | 64 | $16 \times 16$ | $3 \times 3$ | 1 |
| Max-pooling | 64 | $8 \times 8$ | $2 \times 2$ | 2 |
| Convolutional | 64 | $8 \times 8$ | $3 \times 3$ | 1 |
| Convolutional | 64 | $8 \times 8$ | $3 \times 3$ | 1 |
| Max-pooling | 64 | $4 \times 4$ | $2 \times 2$ | 2 |
| Fully-connected | | 200 | $1 \times 1$ | |
| Fully-connected | | #classes | $1 \times 1$ | |

values of parameters based on the results of our preliminary study conducted using the target datasets. For other datasets, we may need to adjust these parameters depending on the complexity of target classes for better performance.

The training data to the CNN is a $65 \times 65$ patch centered at a voxel sample selected at random. We draw 400K voxels divided equally among the classes to avoid the majority class from dominating other classes during training. We use a batch size of 200 and train a total of 60K iterations. After 2K iterations the training set is resampled to learn from diverse examples. We use ADADELTA solver [138] with a momentum of 0.9 and a decay of 0.0005 during training.

The input volume is padded with reflection about the edges before generating patches to accommodate for boundary cases where the sampled voxel is near an edge. Because the structures of interest in the target datasets are rotation-invariant, such padding affects minimally the performance of the CNN.

The CNN trained with balanced classes tends to overestimate the probability of the minority class in the testing stage because of the significant disparity in abundance between the training data, which contain equal amounts of samples among voxel classes, and the testing data, which would have the actual class distribution. Such a class imbalance is increasingly problematic as the disparity grows.

Because the degree of class imbalance in the datasets we used is moderate, a simple way to address this problem is by multiplying the predicted probability of each class by the corresponding prior probability, thereby scaling down the predicted probability of the minority class. We have also implemented the postprocessing technique used in [104], which finds a monotonic cubic polynomial to match the predicted probability and the prior probability. Instead of having a constant scaling factor (i.e. the class prior), this technique transforms the predicted probability in a data-driven way that allows the setting of variable scaling factors with respect to the probability values.

## 7.5 Marching-cubes-based Visualization

We adapt the conventional marching cubes algorithm [130], which extracts isosurfaces from a scalar field (e.g. intensity), to instead extract surfaces based on a high-dimensional feature space. Given an isovalue, the conventional method converts the input scalar field into a binary one using the isovalue as a threshold, which essentially defines the cut-off value of two target structures. The marching cubes algorithm then iterates over the set of cubes, each represented by the values of its eight bounding voxels in the binary scalar field, and accumulates the corresponding

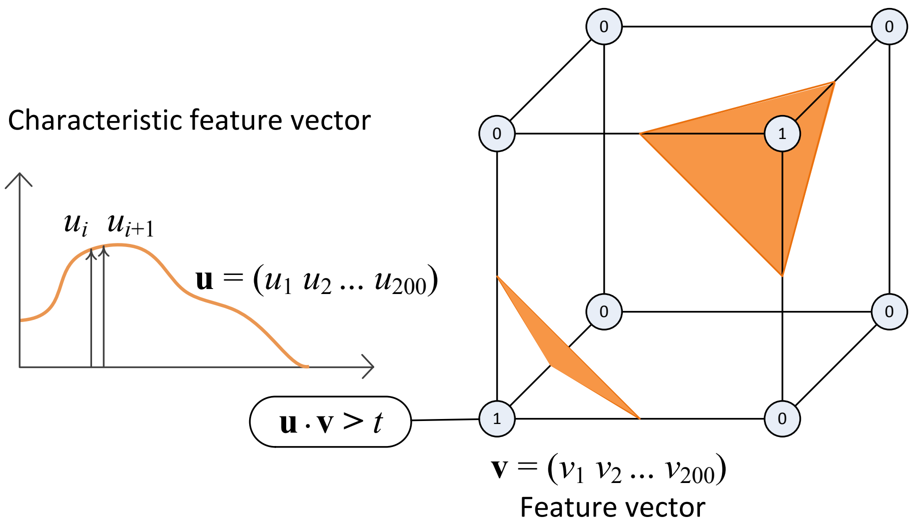

Characteristic feature vector
$u_i$ $u_{i+1}$
$\mathbf{u} = (u_1\ u_2 \ldots u_{200})$
$\mathbf{u} \cdot \mathbf{v} > t$
0
0
0
1
0
0
1
0
$\mathbf{v} = (v_1\ v_2 \ldots v_{200})$
Feature vector

Figure 7.5: Because the high dimensionality of feature space, we generate a binary scalar field by comparing the dot product of the characteristic feature vector $\mathbf{u}$ and the feature vector $\mathbf{v}$ of each voxel with a threshold $t$. After that we apply the conventional marching cubes algorithm on the binarized volume.

triangles based on the eight bits representation. Because in this work the input is a multiscalar field formed by the high-dimensional feature space, we describe in the following a method that converts the input multiscalar field to a binary scalar field with which the marching cubes algorithm can extract surfaces as usual (Figure 7.5).

Assuming that each voxel is represented by a 200-dimensional feature vector $\mathbf{v} = (v_1\, v_2\, \cdots\, v_{200})$, we can decide whether two voxels belong to the same structure by calculating their dot product and compare it with a threshold $t$. Following the same intuition, we can also decide whether a voxel belongs to a user-defined structure, which is specified by a characteristic feature vector $\mathbf{u} = (u_1\, u_2\, \cdots\, u_{200})$. The value of a voxel $\mathbf{v}$ in the binary scalar field, used as the input for the marching cubes algorithm, is calculated as:

$$f(\mathbf{u},\mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{u}\cdot\mathbf{v} \leq t \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

The extracted surface changes with the (binary) value of $f(\mathbf{u},\mathbf{v})$, which is controlled by the characteristic feature vector $\mathbf{u}$ and the value of threshold $t$. Users can interactively configure $\mathbf{u}$ and $t$ as well as the color and opacity of the corresponding surface (used for blending multiple semi-transparent surfaces) using the design widget.

The marching cubes algorithm is highly parallelizable because each cube can be processed independently. Our GPU-based parallel implementation uses the histogram pyramid data structure [139] to allow the query of a triangle's location in the volume and the query of the triangle indices (if any) inside of a specific cube. In a histogram pyramid, each space-partitioning square cell stores the number of triangles in the corresponding partition. The levels from the bottom to the top of the pyramid represent the volumetric data at increasingly lower spatial resolutions. When constructing the pyramid, a cell stores the sum of eight corresponding cells (i.e. $2 \times 2 \times 2$) in the previous (bottom) level. Both queries require a top-down traversal of the pyramid, which only takes a constant time that is logarithmic to the size of spatial dimension (or linear to the height of pyramid). For visual appeal, we smooth the extracted triangular mesh by moving each vertex to the average location of its adjacent vertices in the mesh. During mesh smoothing the query of adjacent vertices is done by first fetching a list of triangles in the current cube and its neighboring cubes (through the second type of queries) and then inspecting each edge of the fetched triangles. We use two vertex buffers to store vertex positions before and after the smoothing. After each smoothing iteration, we swap the two vertex buffers to avoid expensive data movement.

## 7.6 User-mediated Voxel Classification

Using the the 200-dimensional features extracted from the CNN, we will face the same problems in Section 6.4.3, namely the problem of disorganized features and excessive memory usage. Here we address them with similar methods: reordering the abstract features based on feauture-feature similarity and vector quantization.

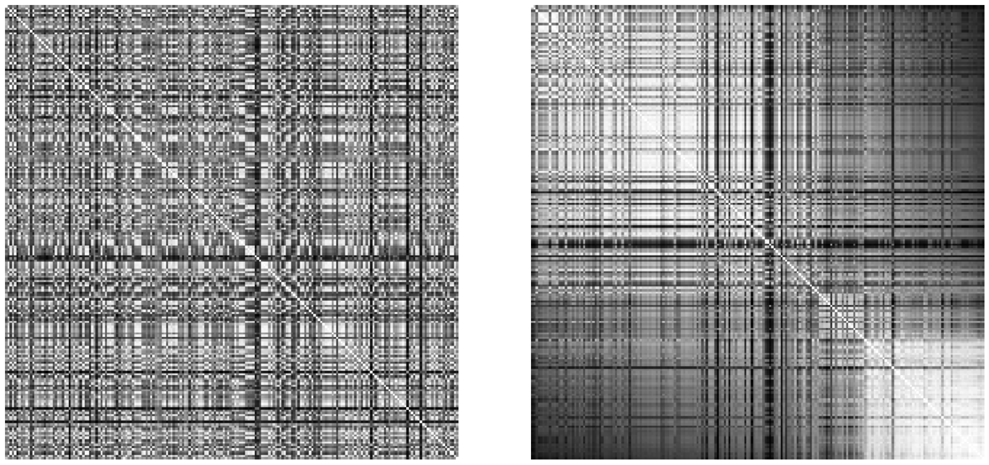
### 7.6.1 Similarity-based Feature Reordering

The order of features in conventional volume visualization designs is usually straightforward and less of a concern because of the low dimensionality of feature spaces. Here we organize the abstract features by their similarities such that users can assign similar weights (e.g. $u_i$ and $u_j$) in the characteristic feature vector to two similar features (e.g. $i$ and $j$-th features).

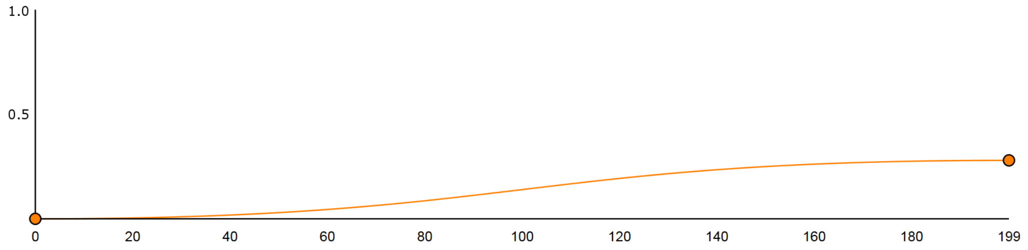
Using the same spectral ordering method in Section 6.4.3, we create a Laplacian matrix $L$ based on feature-feature similarity. Instead of calculating the Fiedler vector of $L$, here we calculate that of the normalized Laplacian matrix $\mathcal{L}$, which is more robust to non-regular graphs [140]; the normalized Laplacian matrix is calculated as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. The rest of the procedure (i.e. reordering features based on the values in the Fiedler vector) remains the same.

We use the following example to show the effect of organizing deep features into the spectral order (Figure 7.6). A bright pixel in the $200 \times 200$ similarity matrix marks a pair of highly correlated features (Figure 7.6a). Many pairs of the 200 features are indeed highly correlated because many pixels are bright. Nevertheless, an arbitrary order of features does not take advantage of such correlations, resulting in a disorganized similarity matrix (left of Figure 7.6a). The spectrally ordered similarity matrix puts similar features closer together, resulting in large bright blocks of various sizes along the diagonal (right of Figure 7.6a). An accessible feature order therefore allows selecting similar voxels using fewer control points in the design widget.

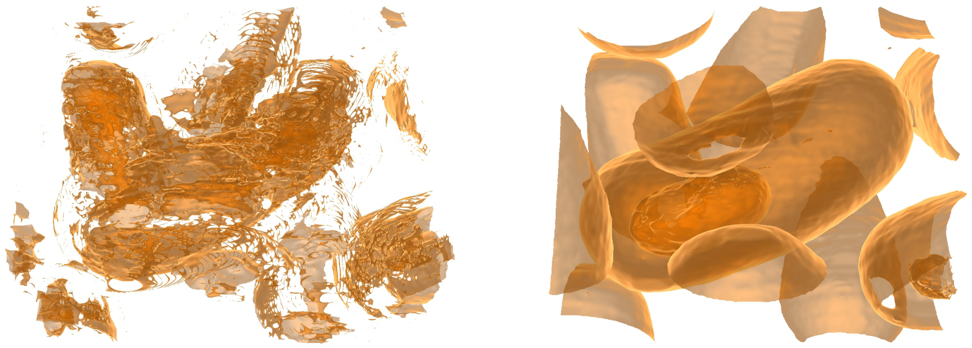
Reordering features into this accessible form is a crucial step. In this example, we use the same characteristic feature vector, which simply specifies an increasing weight from the first (leftmost) to the last (rightmost) feature (Figure 7.6b), and compare the visualization results obtained before and after reordering features. An arbitrary feature order does not exploit the similarity among features and creates surfaces that are broken into small pieces, which do not correspond well to specific structures (left of Figure 7.6c). In contrast, the spectrally ordered features are highly structured such that even a simple characteristic feature vector can reveal some interesting structure (right of Figure 7.6c).



(a) Similarity matrix (Left: arbitrary order. Right: spectrally ordered)



(b) Characteristic feature vector ($t = 0.65$)



(c) Visualization (Left: arbitrary order. Right: spectrally ordered)

Figure 7.6: (a) Before spectral ordering, the similarity matrix, in which a bright pixel represents a pair of highly correlated features, does not show apparent pattern. After spectral ordering, highly correlated features are closer to each other. (b) A simple characteristic feature vector. (c) The arbitrary order of features does not exploit the correlation among features, thus leading to broken surfaces. The spectrally ordered features allow creating a visualization that reveals interesting structures using the same characteristic feature vector.

### 7.6.2 Vector Quantization

Interactive rendering of volumetric data based on high-dimensional voxel representations can be prohibitive because the inflated data size may exceed the memory size on modern GPUs. Directly using 20-dimensional texture features has been shown to be effective previously on small datasets (less than 8.4 million voxels) [124]. Nevertheless, in this paper we use 200-dimensional features and validate our technique on much larger datasets (over one hundred million voxels). Suppose each feature is stored as a four-byte floating point number, loading all 200 features would require 80 gigabytes of memory per hundred million voxels. The high storage demand far exceeds the capability of modern consumer-level graphic cards.

Possible solutions to this problem are out-of-core rendering and data compression. As in Section 6.4.4, we use vector quantization (VQ) to compress the high-dimensional feature vector $\mathbf{v}$ using incremental $k$-means algorithm [120]. Each voxel is encoded by the index of the nearest cluster centroid (among the $k$ code words in the codebook) according to the $L^2$ distance. During rendering, voxels are decoded by referencing the codebook.

Reconstruction using VQ, which is a lossy compression technique, inevitably introduces information loss. The codebook size $k$ is closely related to the reconstruction error of VQ. Choosing a $k$ too small will lead to excessive information loss such that the reconstructed feature vectors (i.e. the cluster centroids found by $k$-means) do not well approximate the actual feature vectors. On the other hand, choosing a $k$ too large will inflate the size of the codebook and require much more

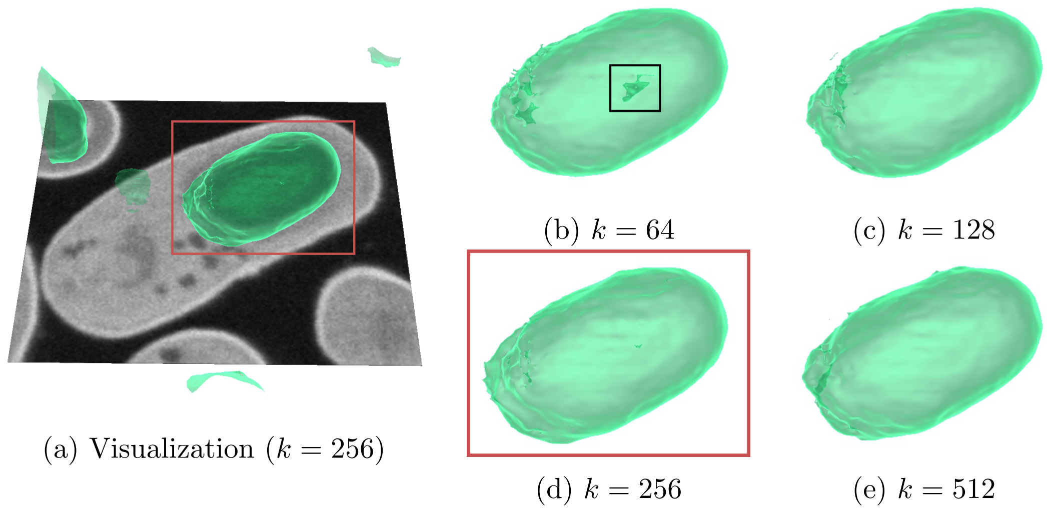


Figure 7.7: The size of codebook $k$ in VQ controls the amount of information loss. The zoom-in views shown in (b–e) are generated from the same subvolume (red box in (a)) in the bacteria dataset with $k$ ranges from 64 to 512. When $k = 64$, the view shows a noticeable disparity near the center of the spore (black box in (b)). The results in (c–e) are comparable without significant visual differences.

computation. A similar information loss introduced by dimension reduction techniques has been shown to be tolerable when simplifying the feature space of volume rendering [127, 131]. In the following, we will evaluate how information loss affects the quality of visualizations created using our deep-learning-assisted approach.

In Figure 7.7 we visually evaluate the quality of the visualization results with various codebook sizes $k$. In this example, we use a subvolume containing a sporulating bacterium of the bacteria dataset (Figure 7.7a). By comparing the results shown in Figure 7.7b–7.7e, the only visually apparent difference is near the center of the spore, where a small region is excluded from the spore, when $k = 64$ (black box in Figure 7.7b); the other three results are comparable despite the difference in $k$ (Figure 7.7c–7.7e). Throughout this work, we choose $k = 256$ such that the code can be stored in eight bits, exactly the same size as the intensity values.

If an outlier voxel is replaced by the centroid of the assigned cluster, calculated as the representative for the majority voxels in that cluster, the error leads to an effect similar to a low-pass filter. These outliers, possibly anomalies or infrequent patterns in the data, will be problematic if they should be presented (instead of filtered out) in the visualization. In our applications that focus on creating a general view of the data, the low-pass filtering effect does not obstruct the understanding and interpretation of the visualization results.

### 7.6.3 Hierarchical Volume Exploration

The extraction of surfaces based on a user-defined characteristic feature vector is flexible in identifying various groups of voxels. Nevertheless, even after reordering features, configuring the characteristic feature vector can still be tedious. In the past, we have used recursive segmentation of intensity-gradient 2D histogram to group similar voxels into a hierarchy [99]. Here we develop a similar semi-automatic method to explore volumetric data by exploiting voxel similarities in high-dimensional feature spaces. This method allows users to efficiently visualize a small set of surfaces in a pre-calculated hierarchy.

Given the result of VQ, we perform binary partition recursively to the adjacency graph based on the $L^2$ distance of the $k$ centroids using spectral clustering [39]. The first binary partition creates two sets of centroids, each corresponds to a group of voxels. Subsequent partitions further subdivide a target set of centroids into two disjoint sets. We organize the partition results into a binary tree, in which a parent node represents the union of the two corresponding groups of voxels represented by the two child nodes. The group of voxels represented by a node defines a surface that can be extracted by the marching cubes algorithm. Based on the binary tree, users can select any nodes to interactively and hierarchically explore the volumetric data. For example, selecting a child node of a parent node shows the subdivision of the surface that corresponds to the parent node.

## 7.7 Results and Discussion

We implemented the CNN using the Caffe framework [121]. The GPU-based parallel implementation of the marching cubes-based volume rendering is implemented using OpenGL and OpenCL[1]. The feature reordering and compression are implemented in MATLAB and Python, respectively. The training time reported in the following is based on a computer with an Intel Xeon 2.6 GHz CPU and an NVIDIA GTX 970 GPU. The training of the CNN took about 2.5 hours; the training needs to be done only once for the training dataset. After training the CNN, the extraction of deep features given new unseen data (i.e. forward pass of a trained CNN) took 3.4 minutes per million voxels.

### 7.7.1 Segmentation Performance Measure

The segmentation performance is evaluated by three established performance measures, namely the VOC score [59], the Rand index [141] and the adjusted Rand index [142]. We also compare the effectiveness of two postprocessing techniques

[1]https://www.khronos.org/opencl/

using these performance measures.

The VOC score is calculated as $\text{VOC} = \frac{\text{TP}}{\text{TP+FP+FN}}$, where TP, FP, and FN denote the number of true positives, false positives, and false negatives, respectively. The VOC score penalizes classifiers with high FP rates, therefore distinguishes reasonable classifiers from classifiers that predict the same class all the time. We calculate class-wise VOC to evaluate the segmentation quality for each class individually.

The Rand index (RI) and adjusted Rand index (ARI) assess segmentation quality based on the portion of agreements among all pairs of $n$ voxels. The RI is calculated as $\text{RI} = \frac{a+b}{\binom{n}{2}}$, where $a$ and $b$ denote the number of voxel pairs that belong to the same and different segments in the ground truth and the generated segmentation. The ARI measures to what extent the segmentation is better than a random segmentation in terms of RI. We calculate both the RI and ARI to evaluate the overall segmentation quality, complementary to the class-wise VOC score.

### 7.7.2 Dataset 1: Bacteria

The first dataset contains one volume of size $1150 \times 450 \times 400$ at resolution 12nm $\times$ 12nm $\times$ 12nm. The original anisotropic volume resolution is 3nm $\times$ 3nm $\times$ 6nm [143]. With this dataset researchers study the sporulation of *B. subtilis* [144], a common rod-shaped gram positive bacterium.

Sporulation (i.e. formation of an endospore) is a unique survival mechanism triggered by certain bacteria in response to environmental stressors such as nutrient depletion. Once initiated, the process of sporulation includes a series of events that are tightly regulated both genetically and temporally. This temporal process results in a sequence of well-defined and replicable morphological states [144], commencing with the asymmetric division of the mother cell, and culminating in the release of a mature spore. The course of cell development is thus of great interest to biologists and is often studied in the model system of B. subtilis.

This dataset is manually labeled with four classes: Resin, cell wall, spore, and others.

1. **Resin**: The dark region in the background.

2. **Cell wall**: The bright thin layer separating the interior of a bacterium and the surrounding resin.

3. **Spore**: The large dark oval-shaped structure inside of a sporulating bacterium. The intensity value and shape of a spore change over time, depending on the life cycle stage of the bacterium. In this dataset, only about 20% of the bacteria are sporulating.

4. **Others**: All the other unlabeled subcellular features (e.g. vesicles and cytoplasm).

Figure 7.8 shows an example of two bacterial cells with the aforementioned structures of interest. Nevertheless, because the bacteria are oriented differently, the appearance of these structures when projecting onto a 2D plane can be significantly different. We divide the original image stack into two sets of equal size

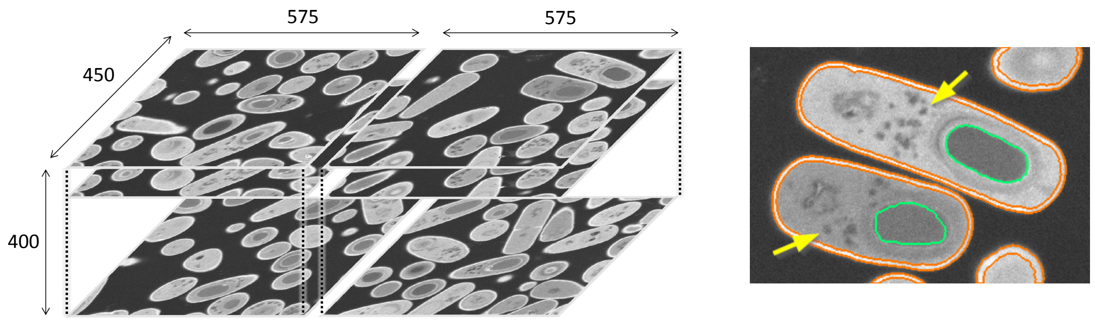


Figure 7.8: (left) The bacteria dataset is divided into the left and right halves for training and testing. (right) Both the spores (green) and the vesicles (yellow arrow) in the two sporulating bacteria are low in intensity. Because the difference in intensity between a spore and other structures (e.g. vesicle and cytoplasm) can be small, conventional intensity-based feature spaces will not differentiate them well.

$575 \times 450 \times 400$, one for training and validation, and the other for testing.

In the following, we evaluate the quality of the segmentation results and compare the effectiveness of the two postprocessing techniques that address the class imbalance problem. In the training volume of the bacteria dataset, the proportion of (resin, cell wall, spore, others) is $(0.472, 0.119, 0.034, 0.374)$, respectively. Because of the scarcity of the spores (0.034), we expect an overestimation of the abundance of spores in the testing stage. Interestingly the result shows that simple postprocessing using class prior can actually deteriorate segmentation performance. Table 7.2 compares the performance measures obtained with and without postprocessing. For the bacteria dataset, using class prior as the scaling factor of predicted probability overestimated the effect of class imbalance and scaled down the predicted probability of minority classes (i.e. spore) too much, resulting in a significant drop in recall for the spore class (from 0.881 to 0.718).

Figure 7.9 shows a concrete example comparing the segmentation results (Figure 7.9c–7.9e) of the input image (Figure 7.9a) with the ground truth (Figure 7.9b).

Table 7.2: The quality measures of the segmentation results for the bacteria and hippocampus datasets. Numbers separated by slashes correspond to different post-processing techniques (none / class prior / fitted transformation).

| | Class | Precision | Recall | VOC |
|---|---|---|---|---|
| Bacteria | Resin | 0.996 / 0.995 / 0.995 | 0.992 / 0.993 / 0.993 | 0.988 / 0.989 / 0.989 |
| | Cell wall | 0.931 / 0.963 / 0.943 | 0.956 / 0.918 / 0.946 | 0.893 / 0.887 / 0.894 |
| | Spore | 0.813 / 0.947 / 0.860 | 0.881 / 0.718 / 0.842 | 0.733 / 0.690 / 0.741 |
| | others | 0.964 / 0.941 / 0.959 | 0.953 / 0.979 / 0.963 | 0.921 / 0.922 / 0.925 |
| | average | 0.970 / 0.970 / 0.971 | 0.969 / 0.969 / 0.971 | 0.943 / 0.941 / 0.945 |
| | | Rand Index (RI) | Adjusted RI (ARI) | |
| | overall | 0.969 / 0.969 / 0.971 | 0.934 / 0.934 / 0.937 | |
| Hippocampus | Non-mito. | 0.999 / 0.994 / 0.991 | 0.959 / 0.989 / 0.993 | 0.958 / 0.983 / 0.985 |
| | Mito. | 0.571 / 0.820 / 0.873 | 0.984 / 0.900 / 0.847 | 0.566 / 0.752 / 0.754 |
| | average | 0.976 / 0.985 / 0.985 | 0.960 / 0.984 / 0.985 | 0.937 / 0.971 / 0.972 |
| | | Rand Index (RI) | Adjusted RI (ARI) | |
| | overall | 0.923 / 0.969 / 0.971 | 0.670 / 0.835 / 0.838 | |

Applying postprocessing using class prior resulted in a significant increase in false negatives (Figure 7.9d) than before postprocessing (Figure 7.9c), therefore lowered the VOC score of the spore class from 0.733 to 0.690. In contrast, the fitted transformation postprocessing is more robust in handling various classes because it derives suitable scaling factors in a data-driven way (Figure 7.9e). Given this flexibility, the fitted transformation increased the VOC scores of all four classes as well as the RI and ARI even though the scores are only marginally better than those without postprocessing.

Having validated the segmentation results, we next present our results on volume visualization using the extracted deep features. We have shown in Section 7.3.1 that conventional feature spaces are ineffective for complex structures. We will now show our deep-learning-assisted approach is effective for complex structures.

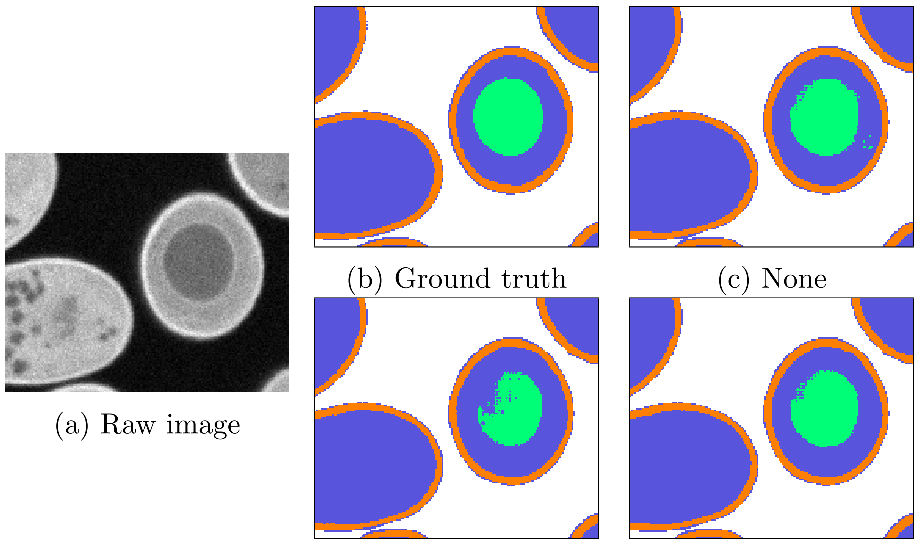
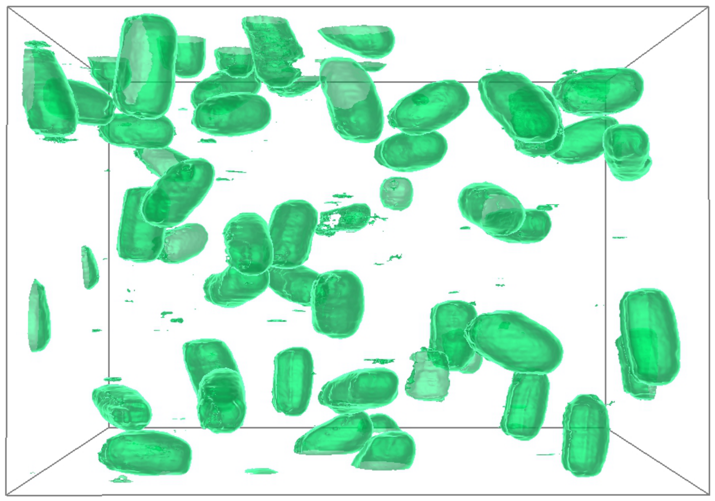
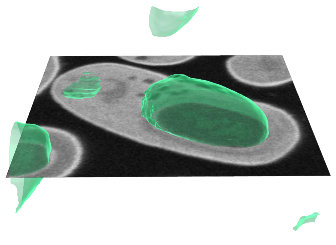

(a) Raw image (b) Ground truth (c) None (d) Class prior (e) Fitted transform

Figure 7.9: This example shows the spore (green) in (b) the ground truth labels and the segmentation results (c) generated without postprocessing, and postprocessed with (d) class prior and (e) fitted transform for the raw image in (a). Simply scaling the predicted probability by class prior resulted in many false negatives for the spore class. In contrast, the data-driven fitted transformation of probability is more robust and generates a result comparable to the ground truth.

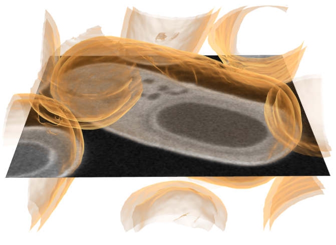
For example, the two characteristic feature vectors shown in Figure 7.10e correspond the surfaces extracted for the spores (Figure 7.10a and Figure 7.10b) and the cell walls (Figure 7.10c), respectively. Combining the two extracted surfaces leads to a composite visualization showing both structures (Figure 7.10d). The close relationships between the semantics and the derived deep features are attributed to the semantic information, in the form of the voxel classes, users provide to the CNN in the training stage. Grouping semantically-related features together in the design widget with spectral ordering facilitates the editing of characteristic feature vectors to target voxels with specific semantics. Further studies are needed to examine the constituents of these semantically-coherent features and whether they can be reused for building visualizations for different target structures.



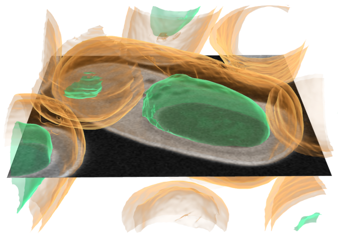
(a) All spores in the bacteria dataset



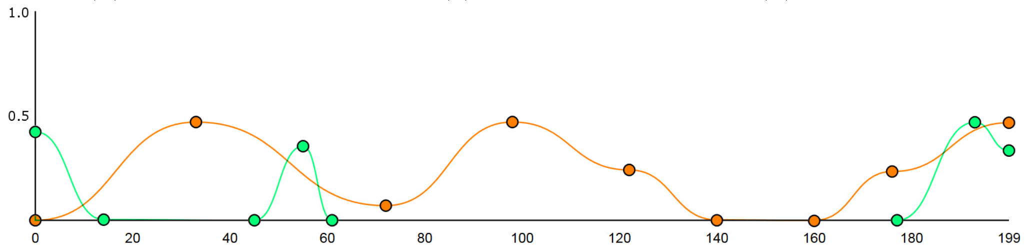
(b) Spore



(c) Cell wall



(d) Composite result



(e) Characteristic feature vectors

Figure 7.10: (a) The visualization of the spores in the bacteria dataset generated using the deep-learning-assisted approach. We show in the next two figures (b) the spore and (c) the cell wall of a bacterial cell. By rendering both semi-transparent surfaces at a time, (d) the composite result shows both the spore and the cell wall. (e) The characteristic feature vectors used to generate the green ($t = 0.36$) and orange ($t = 0.46$) surfaces.

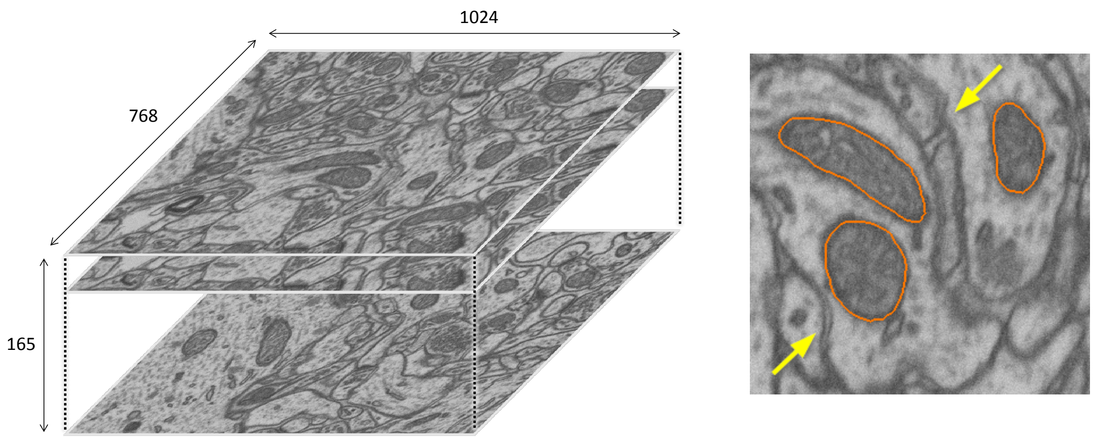


Figure 7.11: (left) The hippocampus dataset consists of two volumes for training and testing (not shown here), both of the same size. (right) Both the mitochondria (orange) and membranes (yellow arrow) are low in intensity, thus making them inseparable using intensity-based features.

## 7.7.3 Dataset 2: Hippocampus

The second dataset[2] [145, 137] contains two volumes of hippocampus, each of size $1024 \times 768 \times 165$ and resolution 5nm $\times$ 5nm $\times$ 5nm. Abnormal changes in morphology and spatial distribution of mitochondria are known to be related to cancer and neurodegenerative diseases such as Parkinson's disease [146]. Better visualization and segmentation of mitochondria are therefore crucial to the study of these diseases. Both volumes in this dataset contain manually-labeled mitochondria (Figure 7.11). All the other structures are considered as non-mitochondria.

Previous studies have shown that contextual features improve segmentation quality over standard features including intensity histogram, gradient magnitude, and texture-related features calculated locally [137]. Similar to the high-level features derived by CNNs, the contextual features describe relationships among voxels

[2] http://cvlab.epfl.ch/data/em

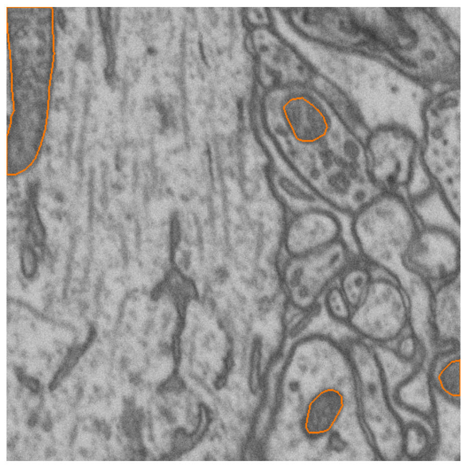
in a large neighborhood, thereby allowing them to depict more precisely the structures of mitochondria. Nevertheless, the contextual features are handcrafted instead of automatically learned.

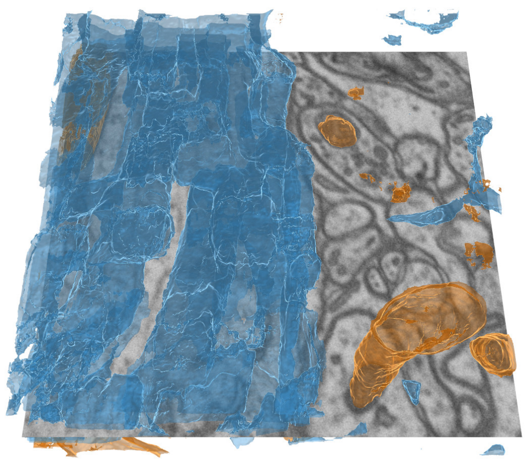
We use the first volume for training and validation, and the second volume for testing. The first volume is divided into the training and the validation sets of size $768 \times 768 \times 165$ and $256 \times 768 \times 165$, respectively.

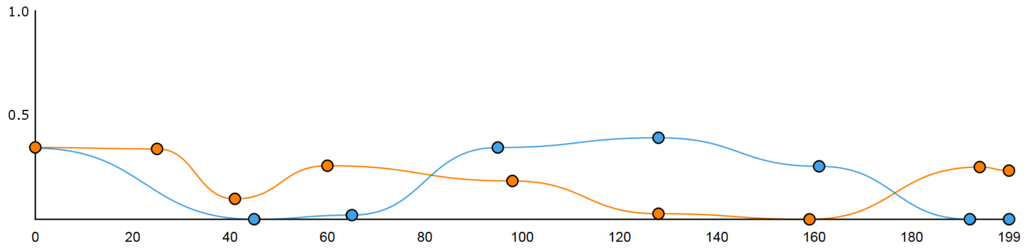
The second part of Table 7.2 shows that the postprocessings are much more influential in addressing the class imbalance problem in the hippocampus dataset, where the mitochondria occupy about four percent of the voxels in the training volume. After applying the two postprocessing techniques, the precision for mitochondria increased significantly whereas the recall decreased only moderately. As a result, the VOC scores of mitochondria increased from 0.566 to 0.752 (class prior) and 0.754 (fitted transformation). Both postprocessing techniques generated VOC scores that are higher than 0.741, which is the state-of-the-art previously reported in [137]. Overall, the RI and ARI both improved substantially after postprocessing.

Compared with the bacteria dataset, the postprocessing has a larger impact on performance in the hippocampus dataset than that in the bacteria dataset. Such a difference occurred because more voxels have similar predicted probabilities for both classes in the hippocampus dataset. After adjusting the predicted probabilities, more voxels in the hippocampus dataset change their predicted class from mitochondria (minority class) to non-mitochondria (majority class).

We have shown that the visualization results of the bacteria dataset (c.f. Figure 7.10) indicate that the derived deep features are semantically meaningful. In that


(a) Raw image


(b) Composite result



(c) Transfer function

Figure 7.12: (a) Within the non-mitochondria region (outside of the orange regions), subregions with noticeably different characteristics can be identified. (b) The extracted deep features enable the detection of the non-mitochondria regions without membranes (blue subvolumes in the left). (c) The two characteristic feature vectors correspond to the mitochondria (orange, t = 63), and the non-mitochondria regions without membranes (blue, t = 49) in the composite visualization.

example, the features align closely with the classes assigned by users. For a voxel class with large intra-class variations, the derived deep features may each detect only a subset of the variants in the same class. For solving a classification problem, the CNN aggregates these features in the last fully-connected layer to detect that specific class as a whole. For creating interesting visualizations, we can use these features targeting different sets of voxels to show subclasses within a user-assigned class.

For example in Figure 7.12b, the mitochondria (orange) and the non-mitochondria regions in the left (blue) correspond to the orange and blue characteristic feature vectors (Figure 7.12c). Although the same (non-mitochondria) class is assigned to both non-mitochondria regions in the left and right during training, the CNN derived different features for them because of their distinct appearance. In the left the single largest non-mitochondria region is bright and uniform, whereas the non-mitochondria regions in the right consist of small bright regions separated by dark membranes. As a result, the blue characteristic feature vector generates only the surface in the left.

### 7.7.4 Dataset 3: Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)

The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS) [61] contains MRI images taken with glioma patients. Here we report results generated with the dataset released with the MICCAI 2013 data challenge[3]. In this dataset, each sample volume consists of four channels, namely the T1, T1c, T2, and Flair channels. All samples in the training set are manually labeled. In clinical practice, physicians decide the tissue types based on the results of multiple channels to diagnose more precisely. The structures of interest in this dataset are the edema, necrosis, non-enhancing and enhancing tumor. The boundaries between abnormal tissues and normal tissues (e.g. the gray matter, the white matter and the cerebrospinal fluid) are usually ambiguous. Figure 7.13 shows an example of the four channels and the four types of abnormal tissues. Typical volumes are cube-shaped,

[3]http://braintumorsegmentation.org/

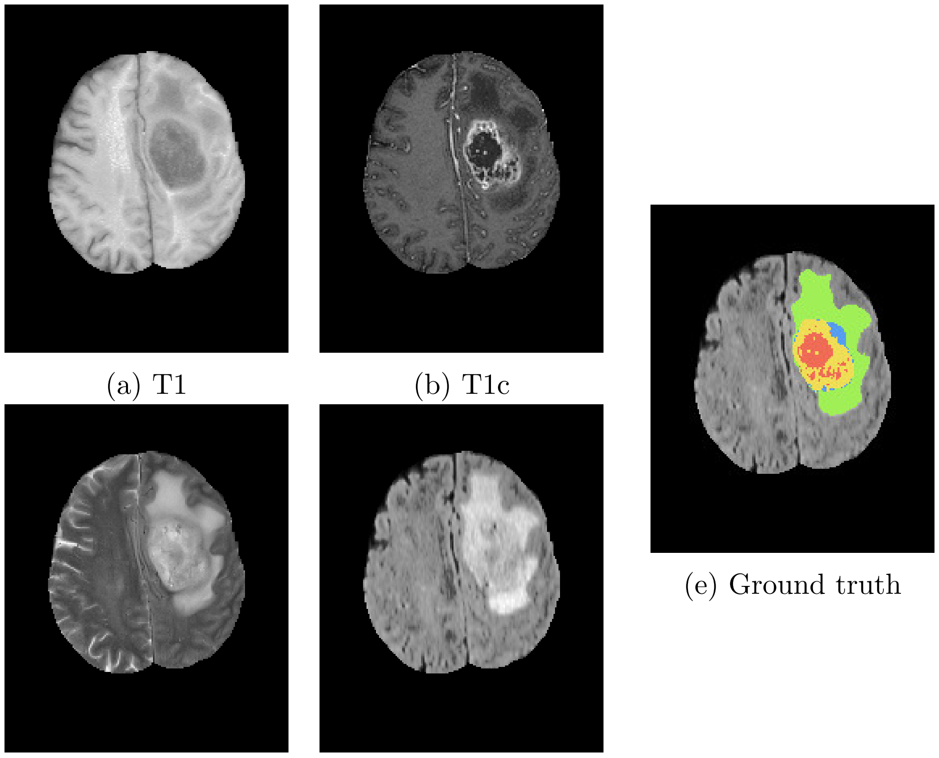

(a) T1 (b) T1c (c) T2 (d) Flair (e) Ground truth

Figure 7.13: A slice in the volume for the (a) T1, (b) T1c, (c) T2, and (d) Flair channels. (e) The ground truth labels of the necrosis (red), edema (green), non-enhancing tumor (blue), and enhancing tumor (yellow).

composed of about six to eight million voxels, resampled and registered into 1mm isotropic resolution.

Here we report the visualization results based on the deep features extracted from a recently published CNN-based method [147], which is designed specifically for the BRATS dataset. Instead of training from scratch, we use the trained CNN model provided by the authors of the original study as a feature extractor that generates 256 deep features[4] for each voxel. We focus on showcasing the application of our visualization design given an alternative high-dimensional feature space, which in this case is also derived by a trained CNN model.

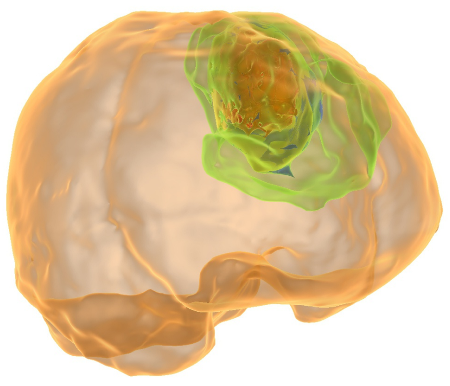
[4]Output of the second fully-connected layer (layer 10 in the original paper).

The visualization presented here is created for a sample with a diagnosis of high-grade gliomas (i.e. HG_0011) in the BRATS dataset (Figure 7.14). Following the hierarchical exploration procedure from the top to the bottom of the tree, the final four surfaces correspond to four leaf nodes in the binary partition tree (Figure 7.14c). Comparing the composite result (Figure 7.14b) created with the four surfaces obtained using the semi-automatic method with the ground truth (Figure 7.14a), we can see some inconsistencies near the center of the tumorous region. For example, the yellow region (enhancing tumor) that wraps around the red region (necrosis) is more apparent in our composite result. These inconsistencies are due to the difficulty in precisely identifying the ambiguous boundaries between tissues of different types.

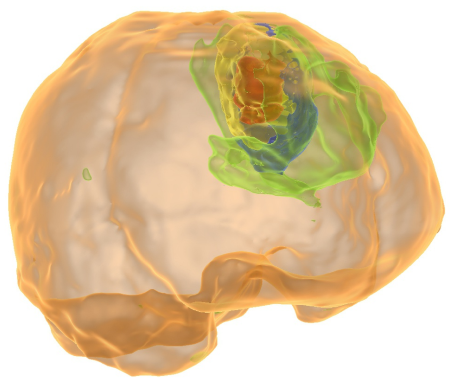
In fact, the BRATS dataset is so challenging that the evaluation of segmentation performance is done after merging multiple voxel classes (e.g. complete tumor for all four classes and core tumor for the three classes excluding edema [61]). We can also flexibly choose to visualize the complete tumor by selecting the purple node (top left corner of Figure 7.14c), which represents the union of the four child subvolumes represented by the leaf nodes.

### 7.7.5 Discussion

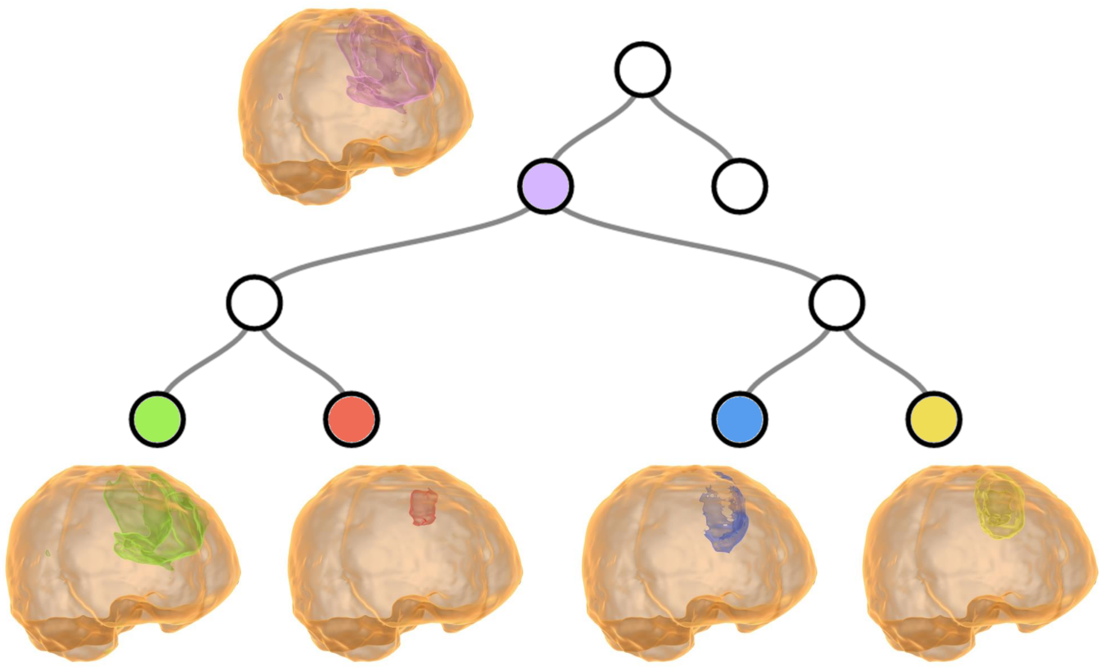
As more and more research applied CNN-based method to this dataset recently (from three in 2014 to nine in 2016), we expect the derived high-level features to be more powerful in the future. Our deep-learning-assisted approach will continue to



(a) Ground truth



(b) Composite result



(c) Nodes and corresponding surfaces

Figure 7.14: (a) The visualization of ground truth labels. (b) The composite result generated using the semi-automatic method. (c) The binary tree allows hierarchical exploration of the volume. The four child subvolumes (represented by the leaf nodes) are created by partitioning the parent subvolume (represented by the purple node).

benefit from the advances in the deep learning methods. Another interesting remark in applying our deep-learning-assisted approach to multimodal data is that the interactions among the four different channels are addressed by the first convolution layer. Therefore, the final deep features we extracted are derived both from a larger context and across various channels. Combining information across channels is crucial to effectively visualize multimodal data. In particular for MRI, using the joint histogram of multiple channels as the feature space has been shown to be effective in showing normal tissues such as cerebrospinal fluid and gray matter [148]. For the tumorous tissues targeted here, the deep features may provide the required context information and descriptive power that lead to better visualizations.

### 7.7.6 Rendering Speed and Computation Time

Our deep-learning-assisted volume visualization introduces extra steps to decode each voxel and create the binary scalar field if compared with the conventional marching cubes algorithm. In our implementation referencing a codebook, which maps the code of a voxel to its corresponding feature vector $\mathbf{v}$, is a simple texture look-up. Updating the binary scalar field requires re-evaluating Equation 7.1, which can be calculated in parallel efficiently by vector arithmetic in the GPU.

We measure the rendering speed with an NVIDIA Quadro K6000 GPU[5]. For a volume with 43 million voxels, our GPU-based volume renderer can render surfaces of similar size (470k triangles) about 45 and 40 frames per second using the conventional marching cubes algorithm and our deep-learning-assisted approach, respectively. Five iterations of smoothing of the mesh with 470k triangles took about 0.26 second, which is too slow if we perform smoothing every time the mesh changes during user interaction. Consequently, we perform mesh smoothing optionally.

Figure 7.15 shows the mean and mean squared error (Figure 7.15a) and the

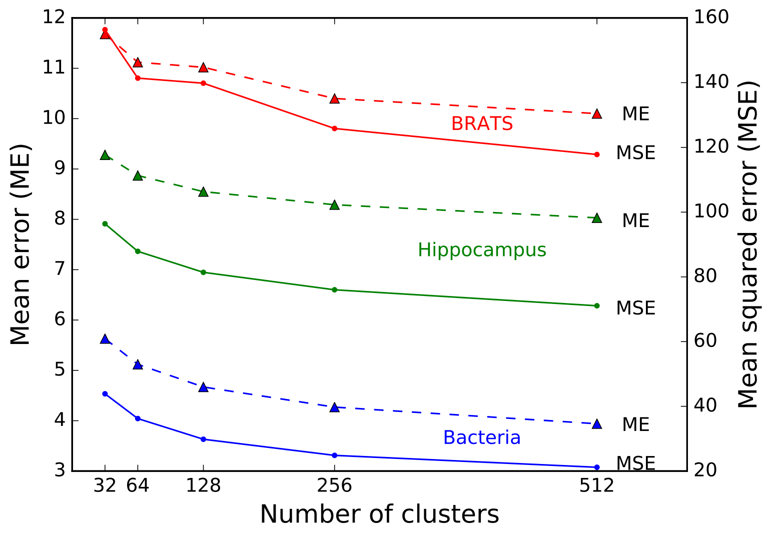
[5]This is different from the NVIDIA GTX 1070 used for training the CNN.

time to perform $k$-means clustering and VQ with $k$ ranging from 32 to 512 (Figure 7.15b). Here we measure error as the $L^2$ distance between the actual and reconstructed vectors. As the value of $k$ increases, both errors decrease exponentially, whereas the running time increases linearly. The reconstruction error for the BRATS dataset is larger than the other two datasets possibly because the BRATS dataset is multimodal with large intra-class variations.

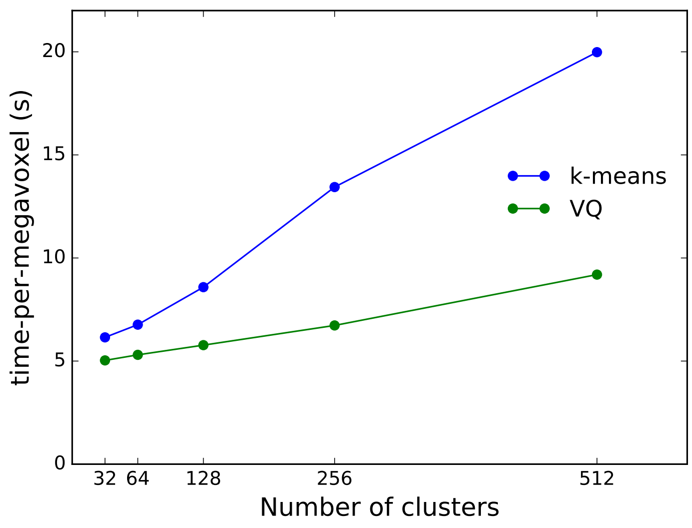
## 7.8 Conclusions and Future Work

Designing volume visualizations is challenging because the current workflow requires users to explicitly define a feasible feature space for the target structures. Existing studies have focused on visualizing structures based on specific handcrafted local features. As the complexity of structures increases, the difficulty of defining a suitable feature space also increases significantly.

In this work, instead of relying on handcrafted features, we use convolutional neural networks (CNNs) to derive automatically useful features from the data. In contrast to the local features that have been used in the past, the features extracted from the CNNs depict high-level concepts that are difficult to describe by local features. We organize the extracted high-dimensional abstract features by spectral ordering such that similar abstract features are close together in the design widget, thereby facilitating interactive exploratory visualization. In addition, we present a semi-automatic technique that creates a binary tree of volume visualizations, which allow users to hierarchically explore volumetric data by traversing the tree.



(a) Reconstruction error



(b) Running Time

Figure 7.15: (a) The reconstruction error of vector quantization decreases as the size of codebook (i.e. number of clusters) increases. (b) Larger codebook requires much more time running $k$-means and VQ.

We have used a simple CNN for feature extraction and voxel-by-voxel segmentation. In the next section, we further improve segmentation performance by applying modern CNN architecture [66] and 3D convolutions to further address the spatial relationships among voxels. In the future, we plan to combine features from different layers in the CNN to design visualizations at various granularities, possibly showing substructures within a structure.

# Chapter 8: CNN-based Segmentation of Volumetric Microscopy Images

## 8.1 Introduction

Recently, convolutional neural networks (CNNs) have been successfully applied to segment natural image datasets such as the Pascal Visual Object Classes (VOC) dataset [59]. Typical training of a CNN requires many labeled samples (e.g. 6929 labeled images in Pascal VOC 2012) that cover a wide variety of the target objects. The diversity of training samples is crucial for the CNNs to learn the actual data distribution. Nevertheless, microscopy image datasets contain considerably fewer labeled samples, usually no more than a few hundred, because of the high costs in collecting and labeling microscopy images. Whereas deeper CNNs have achieved better performance in the past, the scarce training data increasingly raises the risk of overfitting and limited generalization as model complexity grows.

Data augmentations directly address the scarcity of training samples by generating additional similar samples from the existing ones. Standard augmentation operations such as random rotation, scaling, and cropping have been shown to improve prediction accuracy for image classification [60]. Nevertheless, these operations implicitly assume specific properties (e.g. rotation- and scale-invariance) of the target objects in the data; these assumptions are not always realistic or valid in applications other than classifying objects in natural images. Although augmentation methods such as elastic transform have been used for CNN-based segmentation of EM and MRI images [1, 63]. To the best of our knowledge, there has been no systematic evaluation on data augmentations for microscopy image segmentation.

Besides using the application-dependent data augmentations, we approach the problem by finding suitable CNN architectures for volumetric segmentation in general. Both 2D networks, which treat the volume as a stack of image slices and segment slice-by-slice, and 3D networks that directly describe the spatial relationships in all three spatial dimensions reported promising results in the past. One severe drawback of 2D networks is that they only address the spatial relationships within a slice and do not exploit information across slices. On the other hand, the power of 3D convolutions comes at the expense of a significant increase in the number of parameters, which may result in overly complicated networks and potential risk of overfitting. In the past, factorized CNNs that use only low-rank filters instead of full-rank ones were shown to create simplified models that reduce computation, avoids overfitting, and improves generalization for image classification problems [149]. Although factorizations appear to be powerful for applications with scarce training data, their effectiveness in volumetric segmentation is yet to be studied.

Given the strengths and drawbacks for both 2D and 3D networks, in this chapter we study how they compare with each other empirically in terms of segmentation performance for EM datasets. We design and evaluate the performance of 2D and 3D CNNs (Section 8.3) that use factorized convolutions (Section 8.4) and online feature-level augmentations (Section 8.5). We modify the shortcut connection in residual blocks [67, 150] to perform online feature-level augmentations by combining features sampled from a controlled random neighborhood. Finally, we train the CNNs with a Jaccard index-based loss function (Section 8.6) to alleviate the class imbalance problem, which is also aggravated by the limited training data. Our experimental results show that the factorized 3D CNN with online feature-level augmentations performed the best among nine variants of CNN architectures, including the widely used U-Net [1].

## 8.2 Related Work

### 8.2.1 CNN-based Volume Segmentation

The existing CNN-based volume segmentation methods can be grouped into two. The first group of methods [151, 152, 147, 153, 154] train the network to solve a classification problem: determining the class of the center pixel when given a patch of neighboring pixels surrounding it. The surrounding pixels in an input patch offer the context information from which the CNNs can derive useful features for predicting the class for the center pixel. The final segmentation results are created by repeatedly classifying each pixel independently. This problem formulation allows reusing the successful CNN architecture designed for natural image classification–a series of convolution layers with few pooling layers mixed in between, and finally a couple of fully-connected layers to combine various signals into one final prediction [60].

The second group of methods, the fully convolutional networks, recognize that much of the computation is redundant when calculating the convolution for nearby pixels and modify the CNN architecture to reuse intermediate information instead of recalculating it in every independent iteration [66]; this modification accelerates both the training and testing by orders of magnitude. Nevertheless, reusing common information requires including neighboring pixels in the same training iteration; such spatial dependency poses an additional limitation on the diversity of training samples [155]. Because the training data for biomedical applications is already relatively small in practice, further reducing diversity risks overfitting the model.

A typical fully convolutional network consists of an encoding module for learning increasingly higher-level features at a progressively reduced spatial resolution followed by a decoding module for restoring the spatial resolution back (or closer) to the original one [66, 1]. This architecture has been found to be effective and has been adopted and extended in various studies [63, 64, 156]. An alternative to the typical design is omitting the decoding module and make up for the lost spatial resolution at the testing stage by overlapping patches [114].

### 8.2.2 Factorized Convolutions

Learning low-rank filters is an active research topic in computer vision. Replacing 2D convolutions by consecutive 1D convolutions (i.e. vertical then horizontal or horizontal then vertical) shows slightly worse performance with a speed-up factor of 1.6 [157]. Whereas filters that depict useful features may be non-separable, an objective function that penalizes high-rank filters can be used to encourage the learning of low-rank filters [158]. Reconstructing known filters by low-rank approximations shows similar performance with less parameters, effectively simplifying the model [158, 159]. Various ways to approximate known filters were used together with a layer-by-layer fine-tuning method to achieve comparable performance [160]. Instead of approximating known filters, training networks with only low-rank filters can reduce computation, number of parameters, and even improve accuracy [149].

In summary, using low-rank filters in CNNs effectively decreases number of parameters, possibly improving model generalization and preventing overfitting, thus increasing accuracy. Factorized convolutions can therefore benefit applications where training samples are scarce.

### 8.2.3 Augmentations

The most common ways to augment training data for image recognition applications are horizontal reflection and random cropping. A small color variation, calculated from a combination of the first three principal components, is also an effective way to expand the training set if the class of an object is invariant to slight color variations [60]. An elastic transform, which distorts images locally with an offset field, is shown to improve segmentation performance for electron microscopy [1, 63]. Although data augmentations are successful in various applications, they must generate samples that appear to be drawn from the data distribution.

Instead of modifying the training samples, several alternative methods have been proposed to perform augmentations from a different perspective. Fractional pooling that distorts pooling results locally with a stochastic sampling pattern is shown to improve performance [161]. By purposely adding wrongly-labeled training samples, the CNN can be regularized from the loss level [162].

## 8.3 Network Architecture

Our architecture of the CNNs follows the fully convolutional network [66] that takes an input image and outputs the corresponding dense prediction, of the same spatial resolution as the input image. For simplicity, we describe only the 2D version of the CNNs here; for 3D CNNs, replace the relevant components by the 3D counterparts (e.g. 2D by 3D convolutions).

The CNN contains the encoding and the decoding part that gradually decreases and increases spatial resolution, respectively (Figure 8.1). The decoding part, which is shown to be less influential to the overall segmentation performance [163], has much fewer parameters than the encoding part. The CNN consequently has an asymmetric structure as opposed to the symmetric U-Net [1].

We use batch normalization (BN) [6], Parametric Rectified Linear Unit (PReLU) [7], and dropout [8] in the building blocks shown in Figure 8.2. The operations in dashed lines in Figure 8.2b and Figure 8.2c (e.g. downsample, upsample, and $1 \times 1$ convolution) are applied only when the spatial dimension changes to match the spatial

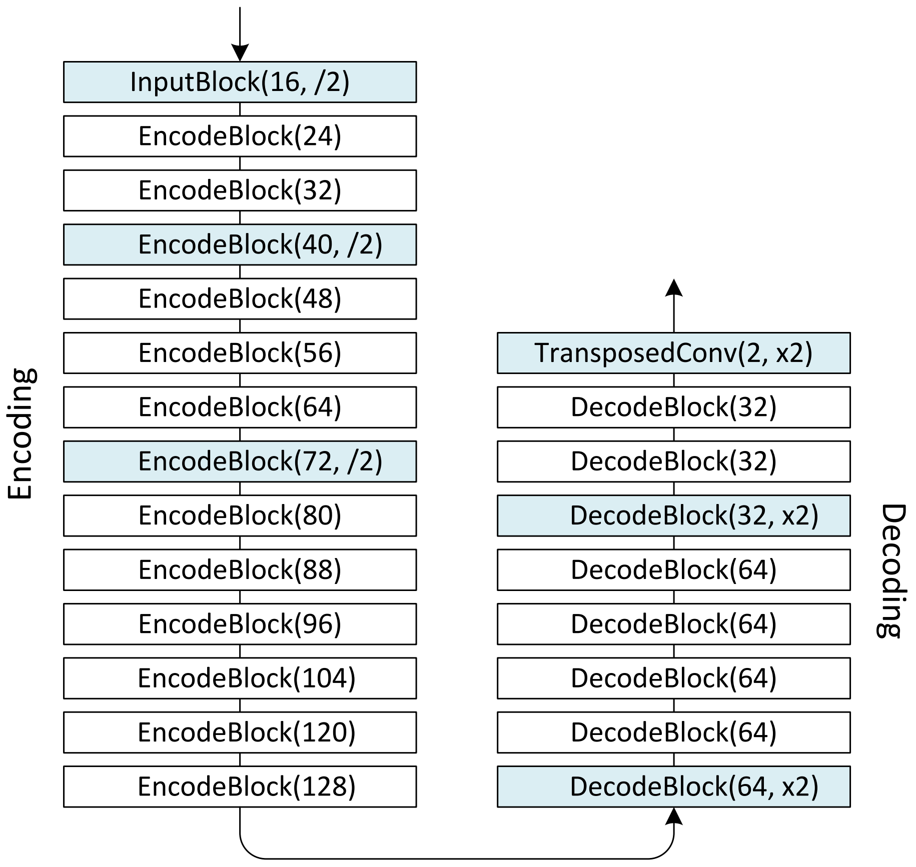
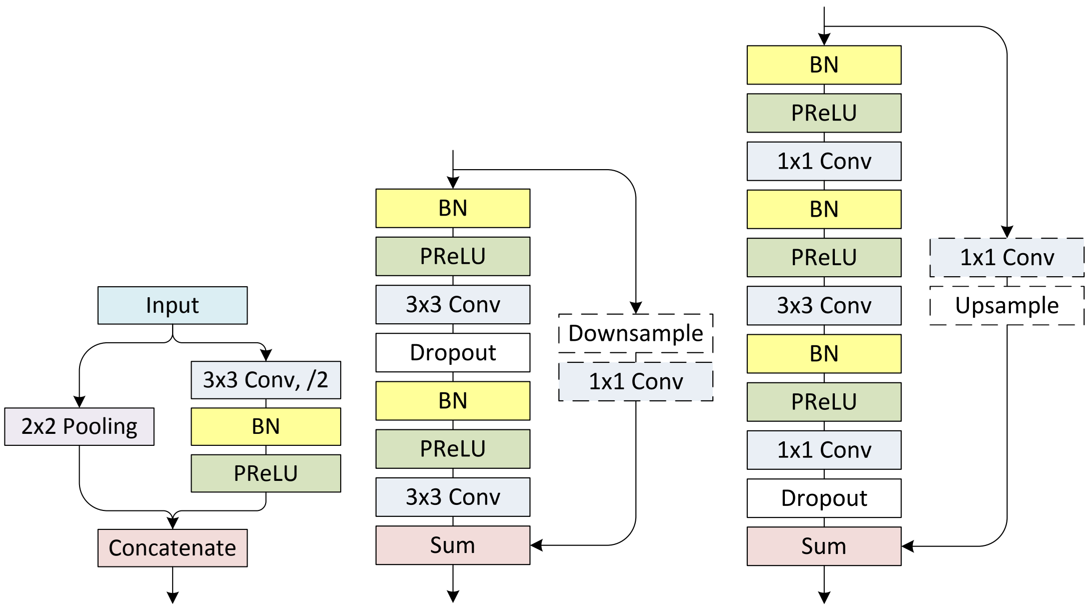


Figure 8.1: The CNN contains the encoding part and the decoding part. The numbers enclosed in the parentheses represent the numbers of output feature maps. A “/2” or “x2” indicates a change in spatial resolution after the (blue) block. See Figure 8.2 for details of each type of blocks.

dimension before summing the feature maps.



(a) Input block (b) Encoding block (c) Decoding block

Figure 8.2: The (a) input block, (b) encoding block, and (c) decoding block used in our CNN. We use batch normalization (BN) [6], Parametric Rectified Linear Unit (PReLU) [7], and dropout [8] in the three building blocks.

Given the input images, we first concatenate the result of max-pooling with the result of stride-two convolution in the input block (Figure 8.2a). The following encoding blocks are implemented as pre-activate residual blocks [150] (Figure 8.2b). In each block, we stack two $3 \times 3$ convolutional layers to enlarge the receptive field with a manageable increase in the number of parameters [109]. The number of feature maps is increased by eight after each encoding block (left column of Figure 6.2) to avoid a sudden increase in the number of feature maps [164]. We use the same zero-padding strategy along the feature dimension [164] for the shortcut connections when the number of feature maps changes (white encoding blocks in Figure 6.2). When spatial resolution decreases (blue encoding blocks in Figure 6.2), the shortcut connections combine features after applying downsampling and linear projection (i.e. $1 \times 1$ convolution). We will elaborate more on modifying the downsampling operation to perform feature-level augmentations later in Section 8.5.

In the decoding part, we use the conventional bottleneck blocks [150], which perform convolutions in a reduced intermediate space created by projection, to further reduce the number of parameters (Figure 8.2c). When spatial resolution increases (blue decoding blocks in Figure 6.2), we replace the $3 \times 3$ convolution by a stride-two $3 \times 3$ transposed convolution [66] to double the spatial resolution.

## 8.4 Factorized Convolution

Although 3D convolutions are powerful in describing the relationships among objects along the $z$-axis, they use 3D kernels that contain an order of magnitude more parameters than 2D kernels. For example a $3 \times 3$ kernel has nine weights whereas a $3 \times 3 \times 3$ kernel has 27 weights. The significant amount of parameters in 3D CNNs may cause the model to overfit the training data and degrade its performance during testing. When only given a limited amount of training data, the overfitting problem is a major concern, especially for the 3D CNNs.

Factorized convolutions that approximate a full-rank 2D kernel with multiple low-rank kernels have been shown to reduce computation without compromising performance [149]. Using low-rank kernels reduces the number of parameters. For example, the total number of parameters decreases from nine to six after replacing a $3 \times 3$ full-rank kernel by a $1 \times 3$ and a $3 \times 1$ rank-1 kernel. Here we replace the $m$ feature maps generated by the $3 \times 3$ full-rank convolution in an encoding block by the concatenation of two rank-1 convolutions, each outputs $\frac{m}{2}$ feature maps. The same strategy is applied to the full-rank 3D kernels.

## 8.5 Feature-level Augmentation

Data augmentation is a common practice to enlarge the training set. Assuming that the target structures are rotation-invariant, we flip the image along the $x$- and $y$-axes (and $z$-axis too in the 3D CNNs) to create more training samples. We also apply elastic deformation [1], which is a commonly used deformation technique for creating realistic tissue images with variations.

Besides the aforementioned data-level augmentations, we modify the residual block to perform online augmentation at the feature-level when downsampling. This idea is motivated by the fractional max-pooling [161], which creates a random spatial sampling pattern to allow non-integer strides when pooling. Because the sampling pattern varies in each iteration, the same input creates a large set of candidate results, distorting the results in all subsequent layers. Here we apply a similar idea and implement the downsampling used in shortcut connections as a random spatial subsampling (Figure 8.3). Whereas the original design adds the input feature maps at specific locations (e.g. all odd rows and odd columns when stride equals two) to the results of convolution [67], we apply a sampling strategy to allow the flexibility of selecting from a local neighborhood. This flexibility introduces augmentations at the feature level and can be used together with other data-level augmentations.

The stochastic sampling pattern is generated by first partitioning each spatial

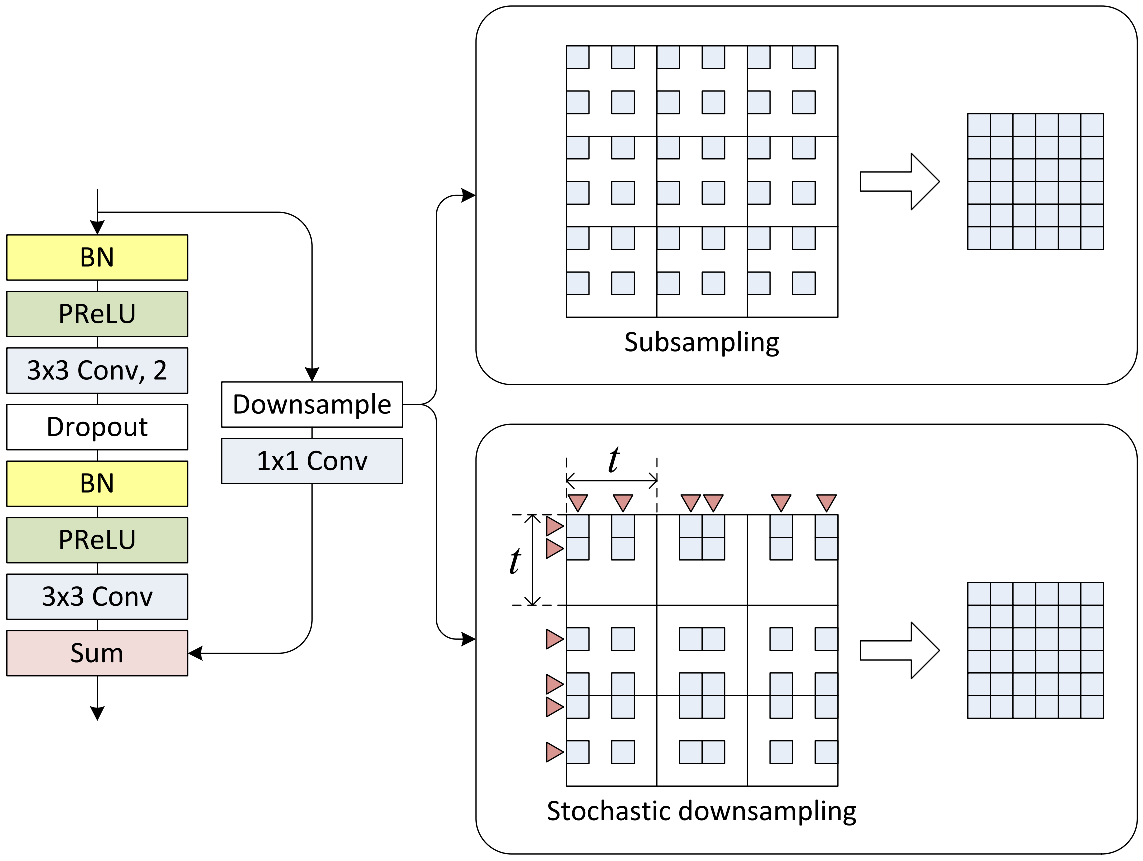
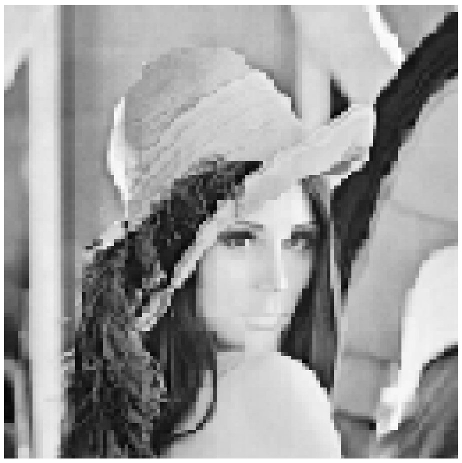


Figure 8.3: We replace the subsampling (top-right) by the stochastic downsampling (bottom-right) when the encoding block reduces spatial resolution. This stochastic operation introduces augmentations at the feature level inside of a residual block.

dimension of the input feature maps into non-overlapping regions of size $t$ and then selecting $\frac{t}{s}$ samples randomly inside of each region (bottom right of Figure 8.3). The samples selected for all spatial dimensions are combined into a complete sampling pattern that outputs feature maps with spatial resolutions identical to that created by subsampling with stride $s$ (top right of Figure 8.3). In fact, subsampling can be regarded as a special case of the stochastic downsampling by fixing the samping pattern. With a fixed stride $s$, the parameter $t$ controls the degree of randomness in the sampling pattern; a larger $t$ leads to a more aggressive augmentation because the summation operation in the residual block can combine more distant neurons.

(a) Conventional subsampling

(b) $t = 4; \mathrm{ME} = 7.11$


(c) $t = 8; \mathrm{ME} = 8.23$

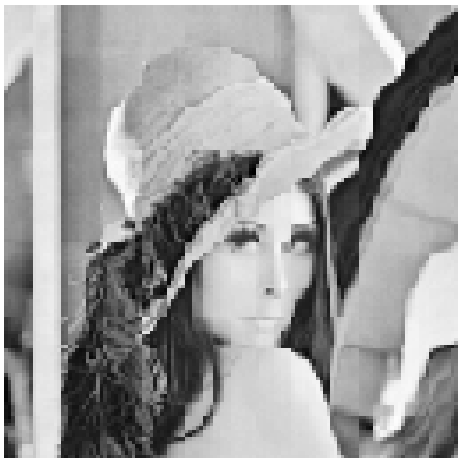

(d) $t = 16; \mathrm{ME} = 11.33$

Figure 8.4: We can control the degree of augmentation by varying the value of $t$. Here we compare the result of (a) conventional subsampling and (b–d) stochastic downsampling with increasing value of $t$. The mean error (ME) comparing (a) and (b–d), caused by the stochasticity, increases from (b) 7.11, (c) 8.23, to (d) 11.33. Pixel values are in the scale of $[0, 255]$.

In this work we choose $t = 4$. During testing, we replace this stochastic operation by an average pooling of the same stride $s$ for a deterministic behavior.

We compare the effect of $t$ by fixing $s$ to two (i.e. reduce spatial resolution by half) in the stochastic downsampling in Figure 8.4. The results in Figure 8.4b–d show increasingly larger distortion as the value of $t$ increases from 4 to 16. If we compare these three images with additional local distortion with the result of subsampling (Figure 8.4a), the mean error also increases with the value of $t$ (from 7.11 to 8.23 and 11.33 for $t = 4, 8, 16$).

## 8.6 Jaccard Index-based Loss

Class imbalance is a severe problem in training with scarce samples of the minority classes. Training a CNN with ordinary loss functions, such as the binary cross-entropy loss, usually ends up with a model that overestimates the prevalence of majority classes (e.g. low precision) and ignores the minority classes (e.g. low recall). Although training with balanced samples followed by postprocessing the prediction probabilities during testing reported promising results in the past [151], this procedure does not exploit the actual class distributions during training and relies on an ad-hoc postprocessing to readjust the probabilities.

In this work, we use an alternative loss function based on Jaccard index, which is also used to measure segmentation performance. For a class $i$, the Jaccard index-based loss $l_{\text{Jacc}}$ is calculated as:

$$l_{\text{Jacc}}(p, y, i) = 1.0 - \frac{\sum_j p_{ij} \cdot y_{ij} + C}{\sum_j p_{ij} + \sum_j y_{ij} - \sum_j p_{ij} \cdot y_{ij} + C}$$

where $p_{ij}$ denotes the predicted probability of the $j$-th voxel belonging to class $i$, and $y_{ij}$ is the one-hot encoding of the ground truth labels that equals one if and only if the $j$-th voxel belongs to class $i$. The constant $C$, which we set to 1.0 in this work, represents a smoothing term and prevents dividing by zero. Because our target segmentation problem is a binary one, we use only the $l_{\text{Jacc}}$ of the minority class (i.e. $l_{\text{Jacc}}(p, y, 1)$ assuming that the minority class is labeled as one) as the final loss function. The average $l_{\text{Jacc}}$ of all classes can be used in multiclass segmentation.

## 8.7 Results and Discussion

We evaluate our CNNs with the same mitochondria dataset [165] (c.f. Section 7.7.3); this dataset contains two fully-labeled volumes, each of size $1024 \times 768 \times 165$, obtained using an electron microscope. The target objects in this dataset are the mitochondria, which are dark objects of oval or elongated shape. The training volume contains only 42 (complete and partial) mitochondria. The CNNs are trained to solve a binary segmentation problem (i.e. mitochondria and non-mitochondria) with a significantly imbalanced class distribution (about 5% mitochondria).

The input samples are random patches of size $256 \times 256$ (2D), extracted from the $xy$-plane, and $128 \times 128 \times 96$ (3D). We ensure that the sampled patches contain mitochondria pixels (or voxels) near the center to avoid training with a minibatch consists of patches that are almost exclusively non-mitochondria pixels (or voxels). All CNNs are trained using the stochastic gradient descent solver with 0.9 momentum and $L_2$ regularization based on the Jaccard index-based loss. We set the batch size to 24 (2D) and three (3D) and train for 30K iterations. The learning rate starts from 0.05 (2D) and 0.1 (3D) and is multiplied by 0.1 at 50% and 75% of the iterations. The dropout rate is set to 0.1. All parameters are set according to our preliminary experiments. We implement all the CNNs using Theano [166] and train them using an NVIDIA GTX1070 GPU with cuDNN library. The training took five

Table 8.1: Segmentation results obtained by U-Net and eight network variants (A–H). The number of parameters is in millions.

| Model | #Parameters | Precision | Recall | Jaccard |
|---|---|---|---|---|
| U-Net [1] | 36.97M | 99.3 / 94.8 | **99.7** / 87.6 | 99.0 / 83.5 |
| A (2D, Full, –) | 1.68M | 99.5 / 91.4 | 99.5 / 91.6 | 99.1 / 84.4 |
| B (2D, Full, A) | 1.68M | 99.5 / 93.0 | 99.6 / 90.8 | 99.1 / 85.0 |
| C (2D, Fact., –) | 0.60M | 99.6 / 91.9 | 99.5 / 93.1 | 99.2 / 86.1 |
| D (2D, Fact., A) | 0.60M | 99.6 / 92.5 | 99.6 / 93.1 | 99.2 / 86.5 |
| E (3D, Full, –) | 4.94M | **99.7** / 92.5 | 99.6 / **93.9** | 99.3 / 87.2 |
| F (3D, Full, A) | 4.94M | 99.6 / 93.2 | 99.6 / 93.4 | 99.3 / 87.5 |
| G (3D, Fact., –) | 0.63M | **99.7** / 92.7 | 99.6 / **93.9** | 99.3 / 87.4 |
| H (3D, Fact., A) | 0.63M | 99.6 / **95.0** | **99.7** / 93.3 | **99.4** / **88.9** |

* The numbers separated by slashes correspond to the results for non-mitochondria (left) and mitochondria (right)
** The best results are marked in bold.

to six hours (2D) and eight to ten hours (3D).

The results in Table 8.1 show the performance of U-Net [1] and eight variants of CNNs. The baseline 2D (model A) and 3D (model E) CNNs that use full-rank kernels and conventional subsampling-based shortcuts are represented by (2D, Full, –) and (3D, Full, –). The variants that use factorized convolutions and online feature-level augmentations are indicated by "Fact." and "A", respectively. We use the Jaccard index of mitochondria as the representative measure for overall segmentation quality.

U-Net is a fully convolutional network with an equal amount of trainable parameters in the encoding and decoding parts. Because of the large numbers of feature maps as well as the symmetric structure, U-Net has significantly more parameters than our baseline 2D CNN (model A). Despite using considerably fewer parameters, model A performed better than U-Net (84.4 vs. 83.5). The superior performance-per-parameter ratio of model A is a result of both the asymmetric structure that focuses on the encoding part and residual learning that uses parameters more effectively than conventional models.

Overall, the 3D CNNs (E–H) showed better segmentation performance than the 2D CNNs (A–D). Even when the numbers of parameters are comparable, the 3D CNNs (G and H) performed much better than the 2D CNNs (C and D). The results suggest that the 2D CNNs are largely limited by the lack of contextual information across the $z$-axis. For example, Figure 8.5 shows the segmentation results of the 80th slice. The 3D CNN (model H) predicted accurately when the 2D CNNs (U-Net and model D) were unsuccessful e.g. false positive near the top-left corner (U-Net and model D) and false negative near the top-right corner (U-Net). This suggested that information along the $z$-axis is especially critical for ambiguous voxels that are difficult to delineate when viewed slice-by-slice.

Our observations for factorizations are consistent with the previous studies [149]: factorized CNNs offer similar (G vs. E) or better performance (C vs. A, D vs. B, and H vs. F) than the non-factorized ones. In fact, in three out of four cases factorization increased the Jaccard index of the foreground class by more than one percent. The results suggest that the factorized CNNs can still learn meaningful high-level features even though each low-rank kernel is less powerful than the full-rank one. The networks with feature-level augmentations performed either slightly better (B vs. A, D vs. C, and F vs. E) or much better (H vs. G) than the ones without such augmentations. The promising results show that feature-level

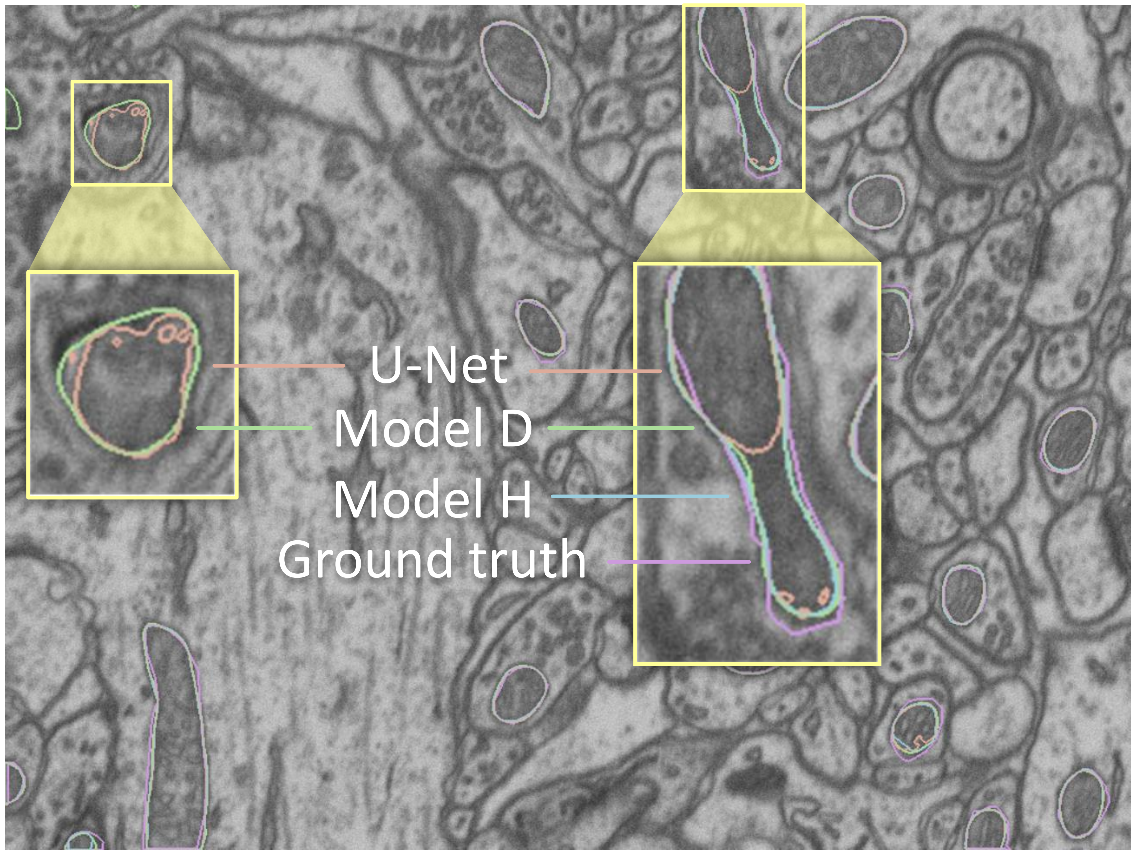


Figure 8.5: This example compares the segmentation results of U-Net (orange), 2D CNN (model D, green), and 3D CNN (model H, blue), with the ground truth (purple). The 3D CNN showed promising result in which the errors are mostly near the ambiguous object boundaries.

augmentations in the residual blocks can introduce a suitable degree of distortion controllable using only one parameter $t$.

We also assess the effect of elastic deformation by training model H without it. Surprisingly, without such deformation the Jaccard indices for both classes (99.4 and 88.8) remain competitive. Because elastic deformation can disrupt the smooth mitochondria boundaries and introduce noise in object shapes, its adverse effect may offset the benefits of augmentations. We will need a specialized augmentation procedure to achieve the full potential of data-level augmentations.

Our best segmentation result (model H) has a VOC score [59] of 94.2. A result of 94.8 is reported in [165], which combines the strengths of kernelized features, extracted by a two-stage training, and subgradient method. In contrast to their sophisticated method, our CNNs are trained end-to-end. Furthermore, many errors made by a machine classifier are due to the inconsistency near the ambiguous object boundaries in the subjective, human-labeled ground truth of this dataset [137]. Figure 8.5 confirms that most errors indeed happened near the boundaries of mitochondria.

## 8.8 Conclusions and Future Work

In this work, we address the challenge of training convolutional neural networks (CNNs) with scarce training data. We compare the performance of various 2D and 3D CNNs trained with limited training samples. Based on the result of our experiments, we found that the 3D CNNs outperformed the 2D ones by a significant margin. Factorized convolutions and online feature-level augmentations improved segmentation performance the most when used together. Our best model (with factorized 3D convolutions and feature-level augmentations) performed significantly better than U-Net [1].

We plan to apply semi-supervised learning and generative models to further address the problem of scarce training samples. We are also interested in applying deep learning approaches for visual knowledge discovery for biomedical datasets [99, 72].

# Chapter 9: Conclusions and Future Work

In this dissertation, we summarized our visual analytic tools for various types of biomedical data. For NetworkViewer, we embed visual representations of protein reaction rules directly inside of the network display. This strategy allows users to always put the relevant rule-specific details in the context of the global network. For Winnow, we visualize the change of disease severity in multiple domains simultaneously and support interactive filtering of patients such that clinicians can interrogate their data visually.

We also studied various ways to visualize biomedical images, addressing computational challenges that arise from the unique characteristics of such images. We developed an efficient segmentation method for gigapixel images, whose results are later used for highlighting texture differences across multiple scales. We later alleviated the need for hand-crafted features and used features derived automatically by a CNN when building visualization. We exploit the similarities among abstract features to facilitate the creation of visualizations based on pixel (voxel) representations in this high-dimensional abstract feature space. We showed promising results when applying similar procedures in creating visualizations for time-lapse images and volumetric images. Finally, we designed CNNs that are suitable for applications where training data is limited. We showed that CNNs that used factorized 3D convolution and feature-level augmentations performed that best among several variants.

Although our visualization tools are designed for specific types of data, such as protein reaction networks and Parkinson's disease clinical data, our tools can be extended to support other data sources with similar characteristics (e.g. network graph and longitudinal data). For example, Winnow can be applied to analyze other chronic diseases.

We used features extracted from a trained CNN for visualization design; the training procedure, however, still requires labeled samples from which the CNN can learn to adjust the parameters. In the future, we can apply unsupervised learning that derives features directly from unlabeled raw data [41, 167, 168, 169, 170]. This way we avoid the time-consuming labeling task, which remains a major bottleneck in many biomedical applications. The pure data-driven, unsupervised approach may, however, create features that do not align with the users' needs. By combining supervised and unsupervised techniques, semi-supervised approaches may derive features suitable for the target task with a minimal number of labeled samples [170].

We approached the problem of scarce training samples from a different perspective because conventional data-level augmentation techniques do not always generate similar variants from existing biomedical images. More recently, generative adversarial networks (GANs) [171, 172] have been applied as generative models for computer vision applications. In fact, a couple of studies have focused on synthesizing biomedical images using GAN [173, 174]. As a possible future direction, we can significantly expand the training set by integrating the generative model into the training procedure, potentially leading to better performance.

# List of Publications

1. Hsueh-Chien Cheng, Antonio Cardone, Somay Jain, Eric Krokos, Kedar Narayan, Sriram Subramaniam, and Amitabh Varshney. Deep-learning-assisted volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, October 2017, accept after minor revision.

2. Eric Krokos, Hsueh-Chien Cheng, Jessica Chang, Celeste Lyn Paul, Bohdan Nebesh, Kirsten Whitley, and Amitabh Varshney. Enhancing deep learning with visual interactions. *ACM Transactions on Interactive Intelligent Systems*, July 2017, submitted.

3. Hsueh-Chien Cheng, Rainer von Coelln, Ann L. Gruber-Baldini, Lisa M. Shulman, and Amitabh Varshney. Winnow: Interactive visualization of temporal changes in multidimensional clinical data. *Journal of Biomedical and Health Informatics*, September 2017, invited to submit to a special issue for best papers in ACM-BCB.

4. Hsueh-Chien Cheng, Rainer von Coelln, Ann L. Gruber-Baldini, Lisa M. Shulman, and Amitabh Varshney. Winnow: Interactive visualization of temporal changes in multidimensional clinical data. In *Proceedings of ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, Boston, MA, August 2017.

5. Hsueh-Chien Cheng, Antonio Cardone, Eric Krokos, Bogdan Stoica, Alan Faden, and Amitabh Varshney. Deep-learning-assisted visualization for live-cell images. In *Proceedings of IEEE International Conference on Image Processing*, Beijing, China, September 2017.

6. Hsueh-Chien Cheng and Amitabh Varshney. Volume segmentation using convolutional neural networks with limited training data. In *Proceedings of IEEE International Conference on Image Processing*, Beijing, China, September 2017.

7. Hsueh-Chien Cheng, Antonio Cardone, and Amitabh Varshney. Interactive exploration of microstructural features in gigapixel microscopy images. In *Proceedings of IEEE International Conference on Image Processing*, Beijing, China, September 2017.

8. Hsueh-Chien Cheng, Bastian R. Angermann, Fengkai Zhang, and Martin Meier-Schellersheim. NetworkViewer: Visualizing biochemical reaction networks with

embedded rendering of molecular interaction rules. *BMC Systems Biology*, 8(1): 70, June 2014.

9. Udayan Khurana, Viet-An Nguyen, Hsueh-Chien Cheng, Jae-wook Ahn, Xi Chen, and Ben Shneiderman. Visual analysis of temporal trends in social networks using edge color coding and metric timelines. In *Proceedings of IEEE International Conference on Social Computing*, pages 549–554, Boston, MA, October 2011. IEEE.

# Bibliography

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[2] Ming-yu Hsieh, Shujie Yang, Mary Ann Raymond-Stinz, Jeremy S Edwards, and Bridget S Wilson. Spatio-temporal modeling of signaling protein recruitment to EGFR. *BMC Systems Biology*, 4:57, 2010.

[3] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[4] Jiangye Yuan, Deliang Wang, and A.M. Cheriyadat. Factorization-based texture segmentation. *IEEE Transactions on Image Processing*, 24(11):3488–3497, November 2015.

[5] Frank G. A. Faas, M. Cristina Avramut, Bernard M. van den Berg, A. Mieke Mommaas, Abraham J. Koster, and Raimond B. G. Ravelli. Virtual nanoscopy: Generation of ultra-large high resolution electron microscopy maps. *The Journal of Cell Biology*, 198(3):457–469, June 2012.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, pages 448–456, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] M. Meier-Schellersheim, X. Xu, B. Angermann, E.J. Kunkel, T. Jin, and R.N. Germain. Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method. *PLoS Computational Biology*, 2(7):e82, 2006.

[10] M. Meier-Schellersheim, I.D.C. Fraser, and F. Klauschen. Multiscale modeling for biologists. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 1(1):4–14, 2009.

[11] Bastian R Angermann, Frederick Klauschen, Alex D Garcia, Thorsten Prustel, Fengkai Zhang, Ronald N Germain, and Martin Meier-Schellersheim. Computational modeling of cellular signaling processes embedded into dynamic spatial contexts. *Nature Methods*, 9(3):283–289, 2012.

[12] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[13] B.J. Breitkreutz, C. Stark, and M. Tyers. Osprey: A network visualization system. *Genome Biology*, 4(3):R22, 2003.

[14] Z. Hu, J. Mellor, J. Wu, and C. DeLisi. VisANT: An online visualization and analysis tool for biological interaction data. *BMC Bioinformatics*, 5(1):17, 2004.

[15] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, Frank T. Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E. Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C. Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B. Kell, Chris Sander, Herbert Sauro, Jacky L. Snoep, Kurt Kohn, and Hiroaki Kitano. The systems biology graphical notation. *Nature Biotechnology*, 27(8):735–741, August 2009.

[16] Kurt W. Kohn, Mirit I. Aladjem, John N. Weinstein, and Yves Pommier. Molecular interaction maps of bioregulatory networks: A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006.

[17] Augustin Luna, Evrim I Karac, Margot Sunshine, Lucas Chang, Ruth Nussinov, Mirit I Aladjem, and Kurt W Kohn. A formal MIM specification and tools for the common exchange of MIM diagrams: An XML-Based format, an API, and a validation method. *BMC Bioinformatics*, 12:167, 2011.

[18] James R. Faeder, Michael L. Blinov, and William S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. In Ivan V. Maly, editor, *Systems Biology*, volume 500, pages 113–167. Humana Press, Totowa, NJ, 2009. ISBN 978-1-934115-64-0 978-1-59745-525-1.

[19] Jérôme Feret, Vincent Danos, Jean Krivine, Russ Harmer, and Walter Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009.

[20] James R. Faeder, Michael L. Blinov, and William S. Hlavacek. Graphical rule-based representation of signal-transduction networks. In *Proceedings of ACM Symposium on Applied Computing*, SAC '05, pages 133–140, New York, NY, USA, 2005. ACM. ISBN 1-58113-964-0.

[21] Bin Hu, G. Matthew Fricke, James R. Faeder, Richard G. Posner, and William S. Hlavacek. GetBonNie for building, analyzing and sharing rule-based models. *Bioinformatics*, 25(11):1457–1460, January 2009.

[22] F. Zhang, B. R. Angermann, and M. Meier-Schellersheim. The Simmune Modeler visual interface for creating signaling networks based on bi-molecular interactions. *Bioinformatics*, 29(9):1229–1230, 2013.

[23] Adam M. Smith, Wen Xu, Yao Sun, James R. Faeder, and G. Elisabeta Marai. RuleBender: Integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, 13(Suppl 8):S3, 2012.

[24] Lily A. Chylek, Bin Hu, Michael L. Blinov, Thierry Emonet, James R. Faeder, Byron Goldstein, Ryan N. Gutenkunst, Jason M. Haugh, Tomasz Lipniacki, Richard G. Posner, Jin Yang, and William S. Hlavacek. Guidelines for visualizing and annotating rule-based models. *Molecular BioSystems*, 7(10):2779–2795, 2011.

[25] Carl-Fredrik Tiger, Falko Krause, Gunnar Cedersund, Robert Palmer, Edda Klipp, Stefan Hohmann, Hiroaki Kitano, and Marcus Krantz. A framework for mapping, visualisation and automatic model creation of signal-transduction networks. *Molecular Systems Biology*, 8:578, April 2012.

[26] Martijn P van Iersel, Alice C Villéger, Tobias Czauderna, Sarah E Boyd, Frank T Bergmann, Augustin Luna, Emek Demir, Anatoly Sorokin, Ugur Dogrusoz, Yukiko Matsuoka, Akira Funahashi, Mirit I Aladjem, Huaiyu Mi, Stuart L Moodie, Hiroaki Kitano, Nicolas Le Novère, and Falk Schreiber. Software support for SBGN maps: SBGN-ML and LibSBGN. *Bioinformatics*, 28(15): 2016–2021, August 2012.

[27] Kurt W Kohn, Mirit I Aladjem, John N Weinstein, and Yves Pommier. Network architecture of signaling from uncoupled helicase-polymerase to cell cycle checkpoints and trans-lesion DNA synthesis. *Cell Cycle*, 8(14):2281–2299, July 2009.

[28] S. North. Drawing graphs with NEATO. http://ftp.graphviz.org/pdf/neatoguide.pdf, April 2004.

[29] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz–Open source graph drawing tools. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Proceedings of International*

*Symposium on Graph Drawing*, Lecture Notes in Computer Science, pages 483–484, Berlin, Heidelberg, January 2002. Springer. ISBN 978-3-540-43309-5 978-3-540-45848-7.

[30] E.R. Gansner, E. Koutsofios, S.C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, March 1993.

[31] Lisa M. Shulman, Robin Leifert Taback, Judy Bean, and William J. Weiner. Comorbidity of the nonmotor symptoms of Parkinson's disease. *Movement Disorders*, 16(3):507–510, 2001.

[32] D. J. Burn, E. N. Rowan, L. M. Allan, S. Molloy, J. T. O'Brien, and I. G. McKeith. Motor subtype and cognitive decline in Parkinson's disease, Parkinson's disease with dementia, and dementia with Lewy bodies. *Journal of Neurology, Neurosurgery, and Psychiatry*, 77(5):585–589, May 2006.

[33] Roxanne E. Jensen, Nan E. Rothrock, Esi M. DeWitt, Brennan Spiegel, Carole A. Tucker, Heidi M. Crane, Christopher B. Forrest, Donald L. Patrick, Rob Fredericksen, Lisa M. Shulman, David Cella, and Paul K. Crane. The role of technical advances in the adoption and integration of patient-reported outcomes in clinical care. *Medical Care*, 53(2):153–159, February 2015.

[34] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM. ISBN 1-58113-084-8.

[35] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[36] Stephanie M. van Rooden, Willem J. Heiser, Joost N. Kok, Dagmar Verbaan, Jacobus J. van Hilten, and Johan Marinus. The identification of Parkinson's disease subtypes using cluster analysis: A systematic review. *Movement Disorders: Official Journal of the Movement Disorder Society*, 25(8):969–978, June 2010.

[37] Ben Shneiderman and Catherine Plaisant. Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 1–7. ACM, 2006.

[38] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[39] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[40] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[41] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[42] William C. Cleveland and Marylyn E. McGill. *Dynamic Graphics for Statistics*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1988. ISBN 978-0-534-09144-6.

[43] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, August 1985.

[44] Amitabh Varshney and Arie Kaufman. FINESSE: A financial information spreadsheet. In *Proceedings of IEEE Symposium on Information Visualization*, pages 70–71. IEEE, 1996.

[45] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of IEEE Symposium on Information Visualization*, pages 89–96, 2004.

[46] Geoffrey Ellis and Alan Dix. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):717–724, 2006.

[47] Hong Zhou, Xiaoru Yuan, Huamin Qu, Weiwei Cui, and Baoquan Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.

[48] J. Jankovic, M. McDermott, J. Carter, S. Gauthier, C. Goetz, L. Golbe, S. Huber, W. Koller, C. Olanow, I. Shoulson, M. Stern, and C. Tanner. Variable expression of Parkinson's disease: A base-line analysis of the DATATOP cohort. *Neurology*, 40(10):1529–1534, October 1990.

[49] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[50] Parkinson Progression Marker Initiative. The Parkinson Progression Marker Initiative (PPMI). *Progress in Neurobiology*, 95(4):629–635, December 2011.

[51] Hans-Christoph Nothdurft. The role of features in preattentive vision: Comparison of orientation, motion and color cues. *Vision Research*, 33(14):1937–1958, September 1993.

[52] Jacob Cohen. Statistical power analysis. *Current Directions in Psychological Science*, 1(3):98–101, June 1992.

[53] Asako Yoritaka, Hideki Ohizumi, Shigeki Tanaka, and Nobutaka Hattori. Parkinson's disease with and without REM sleep behaviour disorder: Are there any clinical differences? *European Neurology*, 61(3):164–170, 2009.

[54] Ivy N. Miller and Alice Cronin-Golomb. Gender differences in Parkinson's disease: Clinical characteristics and cognition. *Movement Disorders*, 25(16): 2695–2703, December 2010.

[55] Robert Patro, Cheuk Yiu Ip, Sujal Bista, Samuel S. Cho, D. Thirumalai, and Amitabh Varshney. MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. In *Proceedings of IEEE Symposium on Biological Data Visualization*, pages 111–118, 2011.

[56] A. Bartesaghi, A. Merk, S. Banerjee, D. Matthies, X. Wu, J. L. S. Milne, and S. Subramaniam. 2.2 {{\AA}} resolution cryo-EM structure of {{$\beta$}}-galactosidase in complex with a cell-permeant inhibitor. *Science*, 348(6239): 1147–1151, June 2015.

[57] Kedar Narayan and Sriram Subramaniam. Focused ion beams in biology. *Nature Methods*, 12(11):1021–1031, October 2015.

[58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[59] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[61] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, Levente Lanczi, Elizabeth Gerstner, Marc-André Weber, Tal Arbel, Brian B. Avants, Nicholas Ayache, Patricia Buendia, D. Louis Collins, Nicolas Cordier, Jason J. Corso, Antonio Criminisi, Tilak Das, Hervé Delingette, Çağatay Demiralp, Christopher R. Durst, Michel Dojat, Senan Doyle, Joana Festa, Florence Forbes, Ezequiel Geremia, Ben Glocker, Polina Golland, Xiaotao Guo, Andac Hamamci, Khan M. Iftekharuddin, Raj Jena, Nigel M. John, Ender Konukoglu, Danial Lashkari, José Antonió Mariz, Raphael Meier, Sérgio Pereira, Doina Precup, Stephen J. Price, Tammy Riklin Raviv, Syed M. S. Reza, Michael Ryan, Duygu Sarikaya, Lawrence Schwartz, Hoo-Chang Shin, Jamie Shotton, Carlos A. Silva, Nuno Sousa, Nagesh K. Subbanna, Gabor Szekely, Thomas J. Taylor, Owen M. Thomas, Nicholas J. Tustison, Gozde Unal, Flor Vasseur, Max Wintermark, Dong Hye Ye, Liang Zhao, Binsheng Zhao, Darko Zikic, Marcel Prastawa, Mauricio Reyes, and Koen Van Leemput. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, October 2015.

[62] Adriënne M Mendrik, Koen L Vincken, Hugo J Kuijf, Marcel Breeuwer, Willem H Bouvy, Jeroen De Bresser, Amir Alansary, Marleen De Bruijne, Aaron Carass, Ayman El-Baz, and others. MRBrainS challenge: Online evaluation framework for brain image segmentation in 3T MRI scans. *Computational Intelligence and Neuroscience*, 2015.

[63] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *arXiv preprint arXiv:1606.04797*, 2016.

[64] Tom Brosch, Lisa Y. W. Tang, Youngjin Yoo, David K. B. Li, Anthony Traboulsee, and Roger Tam. Deep 3D convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. *IEEE Transactions on Medical Imaging*, 35(5):1229–1239, May 2016.

[65] Hao Chen, Xiaojuan Qi, Lequan Yu, and Pheng-Ann Heng. DCAN: Deep contour-aware networks for accurate gland segmentation. *arXiv preprint arXiv:1604.02677*, 2016.

[66] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[68] Mihran Tuceryan and Anil K. Jain. Texture analysis. In *Handbook of Pattern Recognition and Computer Vision*, pages 207–246. World Scientific, 1998.

[69] Cheuk Yiu Ip and Amitabh Varshney. Saliency-assisted navigation of very large landscape images. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1737–1746, 2011.

[70] Roy A. Ruddle, Rhys G. Thomas, Rebecca Randell, Philip Quirke, and Darren Treanor. The design and evaluation of interfaces for navigating gigapixel images in digital pathology. *ACM Transactions on Computer-Human Interaction*, 23 (1):5:1–5:29, January 2016.

[71] Sujal Bista, Ícaro Lins Leitão da Cunha, and Amitabh Varshney. Kinetic depth images: Flexible generation of depth perception. *The Visual Computer*, pages 1–13, May 2016.

[72] Sujal Bista, Jiachen Zhuo, Rao P. Gullapalli, and Amitabh Varshney. Visualization of brain microstructure through spherical harmonics illumination of high fidelity spatio-angular fields. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2516–2525, December 2014.

[73] Sujal Bista, Jiachen Zhuo, Rao P. Gullapalli, and Amitabh Varshney. Visual knowledge discovery for diffusion kurtosis datasets of the human brain. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*, pages 213–234. Springer, Cham, 2015.

[74] P. Bajcsy, A. Vandecreme, J. Amelot, J. Chalfoun, M. Majurski, and M. Brady. Enabling stem cell characterization from large microscopy images. *Computer*, 49(7):70–79, July 2016.

[75] Raphaël Marée, Loïc Rollus, Benjamin Stévens, Renaud Hoyoux, Gilles Louppe, Rémy Vandaele, Jean-Michel Begon, Philipp Kainz, Pierre Geurts, and Louis Wehenkel. Collaborative analysis of multi-gigapixel imaging data using Cytomine. *Bioinformatics*, 32(9):1395–1401, May 2016.

[76] Shaul Hochstein and Merav Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, 36(5):791–804, December 2002.

[77] Robert M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[78] Fumiaki Tomita, Yoshiaki Shirai, and Saburo Tsuji. Description of textures by a structural analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(2):183–191, March 1982.

[79] M. Tuceryan and AK. Jain. Texture segmentation using Voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):211–216, February 1990.

[80] B.B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):72–77, January 1995.

[81] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using Gabor filters. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pages 14–19, 1990.

[82] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, November 1995.

[83] Adrien Depeursinge, Antonio Foncubierta-Rodriguez, Dimitri Van de Ville, and Henning Müller. Multiscale lung texture signature learning using the Riesz transform. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 15, pages 517–524, 2012.

[84] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[85] T. Ojala, M. Pietikäinen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[86] S. Liao, M.W.K. Law, and A.C.S. Chung. Dominant local binary patterns for texture classification. *IEEE Transactions on Image Processing*, 18(5):1107–1118, 2009.

[87] Zhenhua Guo, L. Zhang, and D. Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing*, 19(6):1657–1663, 2010.

[88] Xiaoyang Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650, 2010.

[89] Jianfeng Ren, Xudong Jiang, and Junsong Yuan. Noise-resistant local binary pattern with an embedded error-correction mechanism. *IEEE Transactions on Image Processing*, 22(10):4049–4060, October 2013.

[90] K. Mosaliganti, A. Gelas, A. Gouaillard, and S. Megason. Tissue level segmentation and tracking of biological structures in microscopic images based on density maps. In *Proceedings of IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1354–1357, 2009.

[91] Tse-Wei Chen, Yi-Ling Chen, and Shao-Yi Chien. Fast image segmentation based on K-Means clustering with histograms in HSV color space. In *Proceedings of IEEE Workshop on Multimedia Signal Processing*, pages 322–325, 2008.

[92] Zhiding Yu, Oscar C. Au, Ruobing Zou, Weiyu Yu, and Jing Tian. An adaptive unsupervised approach toward pixel clustering and color image segmentation. *Pattern Recognition*, 43(5):1889–1906, 2010.

[93] C.H.Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and H.D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of IEEE International Conference on Data Mining*, pages 107–114, 2001.

[94] Horst Wildenauer, Branislav Mičušík, and Markus Vincze. Efficient texture representation using multi-scale regions. In *Proceedings of Asian Conference on Computer Vision*, pages 65–74, 2007. ISBN 978-3-540-76385-7 978-3-540-76386-4.

[95] A Levinshtein, S. Dickinson, and C. Sminchisescu. Multiscale symmetric part detection and grouping. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2162–2169, September 2009.

[96] Cevahir Cigla and A. Aydin Alatan. Efficient graph-based image segmentation via speeded-up turbo pixels. In *Proceedings of IEEE International Conference on Image Processing*, pages 3013–3016, 2010.

[97] R. Achanta, A Shaji, K. Smith, A Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, November 2012.

[98] Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling.* Sage Publications, Beverly Hills, 1978. ISBN 978-0-8039-0940-3.

[99] Cheuk Yiu Ip, Amitabh Varshney, and Joseph JaJa. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2355–2363, 2012.

[100] S. Yi, I. Yoon, C. Oh, and Y. Yi. Real-time integrated face detection and recognition on embedded GPGPUs. In *Proceedings of IEEE Symposium on Embedded Systems for Real-Time Multimedia*, pages 98–107, October 2014.

[101] David J. Loane, Alok Kumar, Bogdan A. Stoica, Rainier Cabatbat, and Alan I. Faden. Progressive neurodegeneration after experimental brain trauma: Association with chronic microglial activation. *Journal of Neuropathology and Experimental Neurology*, 73(1):14–29, January 2014.

[102] Bogdan A. Stoica, David J. Loane, Zaorui Zhao, Shruti V. Kabadi, Marie Hanscom, Kimberly R. Byrnes, and Alan I. Faden. PARP-1 inhibition attenuates neuronal loss, microglia activation and neurological deficits after traumatic brain injury. *Journal of Neurotrauma*, 31(8):758–772, April 2014.

[103] Shruti V. Kabadi, Bogdan A. Stoica, David J. Loane, Tao Luo, and Alan I. Faden. CR8, a novel inhibitor of CDK, limits microglial activation, astrocytosis, neuronal loss, and neurologic dysfunction after experimental traumatic brain injury. *Journal of Cerebral Blood Flow and Metabolism*, 34(3):502–513, March 2014.

[104] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, June 2012.

[105] Dan C. Cireşan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 411–418. Springer, 2013.

[106] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, January 2013.

[107] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[108] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, September 2014.

[109] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[110] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.

[111] Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.

[112] Kisuk Lee, Aleksandar Zlateski, Ashwin Vishwanathan, and H. Sebastian Seung. Recursive training of 2D-3D convolutional networks for neuronal boundary detection. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, NIPS'15, pages 3573–3581, Cambridge, MA, USA, 2015. MIT Press.

[113] Qi Dou, Hao Chen, Lequan Yu, Lei Zhao, Jing Qin, Defeng Wang, Vincent CT Mok, Lin Shi, and Pheng-Ann Heng. Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks. *IEEE Transactions on Medical Imaging*, 35(5):1182–1195, May 2016.

[114] Konstantinos Kamnitsas, Christian Ledig, Virginia F.J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, and Ben Glocker. Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, February 2017.

[115] Abhishek Sharma, Oliver Grau, and Mario Fritz. VConv-DAE: Deep volumetric shape learning without object labels. In *Proceedings of European Conference on Computer Vision Workshops*, pages 236–250. Springer, Cham, October 2016.

[116] Yueqing Wang, Zhige Xie, Kai Xu, Yong Dou, and Yuanwu Lei. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. *Neurocomputing*, 174, Part B:988–998, January 2016.

[117] Kang Li, Mei Chen, and Takeo Kanade. Cell population tracking and lineage construction with spatiotemporal context. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 295–302. Springer, Berlin, Heidelberg, October 2007.

[118] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, March 2015.

[119] Bojan Mohar. Laplace eigenvalues of graphs—a survey. *Discrete Mathematics*, 109(1):171–183, November 1992.

[120] David Sculley. Web-scale k-means clustering. In *Proceedings of the International Conference on World Wide Web*, pages 1177–1178. ACM, 2010.

[121] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014.

[122] Carlos Correa and Kwan-Liu Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380–1387, 2008.

[123] S. Wesarg, M. Kirschner, and M. F. Khan. 2D histogram based volume visualization: Combining intensity and size of anatomical structures. *International Journal of Computer Assisted Radiology and Surgery*, 5(6):655–666, 2010.

[124] J.J. Caban and P. Rheingans. Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1364–1371, November 2008.

[125] Carlos Correa and Kwan-Liu Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):192–204, February 2011.

[126] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[127] Xin Zhao and Arie E. Kaufman. Multi-dimensional reduction and transfer function design using parallel coordinates. In *Proceedings of IEEE International Conference on Volume Graphics*, pages 69–76, 2010.

[128] R. Maciejewski, Yun Jang, Insoo Woo, H. Jänicke, K.P. Gaither, and D.S. Ebert. Abstracting attribute space for transfer function exploration and design. *IEEE Transactions on Visualization and Computer Graphics*, 19(1):94–107, January 2013.

[129] Ross Maciejewski, Insoo Woo, Wei Chen, and David S. Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473–1480, 2009.

[130] William E. Lorensen and Harvey E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM. ISBN 978-0-89791-227-3.

[131] Han Suk Kim, Jürgen P Schulze, Angela C Cone, Gina E Sosinsky, and Maryann E Martone. Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. *Information Visualization*, 9(3): 167–180, 2010.

[132] J. Kruger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Conference on Visualization*, pages 287–292, Washington, DC, USA, 2003. ISBN 978-0-7695-2030-8.

[133] Issei Fujishiro, Yuji Maeda, Hiroshi Sato, and Yuriko Takeshima. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):144–155, 1996.

[134] P. Bhaniramka, Caixia Zhang, Daqing Xue, R. Crawfis, and R. Wenger. Volume interval segmentation and rendering. In *Proceedings of IEEE Symposium on Volume Visualization and Graphics*, pages 55–62, October 2004.

[135] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Conference on Visualization*, pages 513–520, 2003.

[136] Thomas Gerstner. Multiresolution extraction and rendering of transparent isosurfaces. *Computers & Graphics*, 26(2):219–228, April 2002.

[137] Aurélien Lucchi, Carlos Becker, Pablo Márquez Neila, and Pascal Fua. Exploiting enclosing membranes and contextual cues for mitochondria segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 65–72, 2014.

[138] Matthew D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701 [cs]*, December 2012.

[139] Christopher Dyken, Gernot Ziegler, Christian Theobalt, and Hans-Peter Seidel. High-speed marching cubes using HistoPyramids. *Computer Graphics Forum*, 27(8):2028–2039, December 2008.

[140] Fan RK Chung. *Spectral Graph Theory*, volume 92. American Mathematical Soc., 1997.

[141] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[142] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

[143] Kedar Narayan, Cindy M Danielson, Ken Lagarec, Bradley C Lowekamp, Phil Coffman, Alexandre Laquerre, Michael W Phaneuf, Thomas J Hope, and Sriram Subramaniam. Multi-resolution correlative focused ion beam scanning electron microscopy: Applications to cell biology. *Journal of Structural Biology*, 185(3): 278–284, March 2014.

[144] Irene S. Tan and Kumaran S. Ramamurthi. Spore formation in Bacillus subtilis. *Environmental Microbiology Reports*, 6(3):212–225, June 2014.

[145] A. Lucchi, Yunpeng Li, and P. Fua. Learning for structured prediction using approximate subgradient descent with working sets. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1987–1994, June 2013.

[146] Angela C. Poole, Ruth E. Thomas, Laurie A. Andrews, Heidi M. McBride, Alexander J. Whitworth, and Leo J. Pallanck. The PINK1/Parkin pathway regulates mitochondrial morphology. *Proceedings of the National Academy of Sciences of the United States of America*, 105(5):1638–1643, February 2008.

[147] Sergio Pereira, Adriano Pinto, Victor Alves, and Carlos A. Silva. Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, May 2016.

[148] Joe Kniss, Jürgen P. Schulze, Uwe Wössner, Peter Winkler, Ulrich Lang, and Charles Hansen. Medical applications of multi-field volume rendering and VR techniques. In *Proceedings of the Joint Eurographics - IEEE Conference on Visualization*, VISSYM'04, pages 249–254, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN 978-3-905673-07-4.

[149] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training Convolutional Neural Networks with Low-Rank Filters for Efficient Image Classification. In *Proceedings of International Conference on Learning Representations*, 2016.

[150] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of European Conference on Computer Vision*, pages 630–645, 2016.

[151] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, pages 2843–2851, 2012.

[152] Alexander Brebisson and Giovanni Montana. Deep neural networks for anatomical brain segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–28, 2015.

[153] Pim Moeskops, Max A. Viergever, Adrienne M. Mendrik, Linda S. de Vries, Manon J. N. L. Benders, and Ivana Isgum. Automatic segmentation of MR brain images with a convolutional neural network. *IEEE Transactions on Medical Imaging*, 35(5):1252–1261, May 2016.

[154] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, January 2017.

[155] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. PixelNet: Towards a general pixel-level architecture. *arXiv preprint arXiv:1609.06694*, 2016.

[156] Hao Chen, Qi Dou, Lequan Yu, and Pheng-Ann Heng. VoxResNet: Deep voxelwise residual networks for volumetric brain segmentation. *arXiv preprint arXiv:1608.05895*, 2016.

[157] Franck Mamalet and Christophe Garcia. Simplifying ConvNets for fast learning. In *Proceedings of International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science, pages 58–65. Springer, Berlin, Heidelberg, September 2012. ISBN 978-3-642-33265-4 978-3-642-33266-1.

[158] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2754–2761, 2013.

[159] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*, May 2014.

[160] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, NIPS'14, pages 1269–1277, Cambridge, MA, USA, 2014. MIT Press.

[161] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[162] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. DisturbLabel: Regularizing CNN on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[163] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[164] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep Pyramidal Residual Networks. *arXiv preprint arXiv:1610.02915*, 2016.

[165] Aurélien Lucchi, Pablo Márquez-Neila, Carlos Becker, Yunpeng Li, Kevin Smith, Graham Knott, and Pascal Fua. Learning structured models for segmentation of 2-D and 3-D imagery. *IEEE Transactions on Medical Imaging*, 34 (5):1096–1110, 2015.

[166] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, and others. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[167] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[168] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of International Conference on Learning Representations*, 2014.

[169] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of International Conference on Machine Learning*, pages 1558–1566, 2016.

[170] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. In *Proceedings of International Conference on Learning Representations*, 2016.

[171] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, pages 2672–2680, June 2014.

[172] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of International Conference on Learning Representations*, November 2015.

[173] Dong Nie, Roger Trullo, Caroline Petitjean, Su Ruan, and Dinggang Shen. Medical image synthesis with context-aware generative adversarial networks. *arXiv:1612.05362 [cs]*, December 2016.

[174] Pedro Costa, Adrian Galdran, Maria Inês Meyer, Michael David Abràmoff, Meindert Niemeijer, Ana Maria Mendonça, and Aurélio Campilho. Towards adversarial retinal image synthesis. *arXiv preprint arXiv:1701.08974*, 2017.