Visualization Techniques for Neuroscience-Inspired Dynamic Architectures

Margaret Grace Drouhard
University of Tennessee - Knoxville, mwhite40@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Margaret Grace Drouhard entitled "Visualization Techniques for Neuroscience-Inspired Dynamic Architectures." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

J. Douglas Birdwell, Major Professor

We have read this thesis and recommend its acceptance:

Mark E. Dean, Chad Steed

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Visualization Techniques for Neuroscience-Inspired Dynamic Architectures

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Margaret Grace Drouhard

May 2015

For Patrick, Neven, and Trevor. May you always retain your fascination for building, sketching, and creating.

Acknowledgements

I would like to thank my advisor, Dr. Doug Birdwell, for bringing me into an impactful research group, and for constantly challenging me and providing constructive feedback. I would like to thank Dr. Mark Dean, another of my committee members and research group leaders, for his insight, support, and passion for improving Computer Science and Engineering education. I would like to thank Dr. Chad Steed for sparking my interest in visualization with his excellent InfoVis course, and for mentoring me in my thesis and other research projects, as well as serving on my committee. I would like to thank Dr. Jens Gregor for inspiring me with intellectual challenges, especially in my early C.S. education, and for allowing me to serve as his teaching assistant for two years. I would like to thank Dr. Lynne Parker for the mentorship and encouragement she offered me, and for serving as such an excellent role model for women in computing.

I would like to thank the other faculty and students in the neuromorphic computing research group who have provided invaluable feedback on this work: Dr. Roger Horn, Dr. Tsewei Wang, Luke Bechtel, Jason Chan, Chris Daffron, Dr. Catherine Schuman, Ryan Wagner, and Josh Willis. I would like to thank the original Systers founders, Sadika Amreen, Denise Gosnell, Zahra Mahoor, Casey Miller, Nicole Pennington, and Katie Schuman, for their commitment to diversifying and constantly improving the field for women in technology. I would like to thank all of my other mentors, peers, and study buddies within the Electrical Engineering and Computer Science (EECS) Department. I would like to thank Randy Bond, Dana

Bryson, Bryan Burke, Kim Cox, Julia Elkins, Sonja Hill, Markus Iturriaga Woelfel, Melanie Kelley, Linda Robinson, and the EECS office and IT staff for their hard work for the department, particularly in supporting graduate students.

I would like to thank my family and friends for always supporting me, even when what I was doing didn't seem to make sense. I would especially like to thank Kevin Drouhard, Vicky White, Kate Kerekes, Molly White, Ryan Kerekes, Melanie Drouhard, Julie Buonocore, Ann Huston, Sesha Pinnaduwage, and Liz Thurman. I would like to thank the Roadkill Running Group and Sheer Inspiration Pole Family for helping for helping me sustain my physical and emotional health through the ups and downs of grad school.

Abstract

This work introduces visualization tools for Neuroscience-Inspired Dynamic Architecture (NIDA) networks and for the Dynamic Adaptive Neural Network Array (DANNA) hardware implementation of NIDA. A NIDA network is a novel type of artificial neural network that has performed well on control, anomaly detection, and classification tasks. We introduce a three dimensional visualization of software NIDA networks that represents network structure and simulates activity on networks. We present some of the analysis tasks for which the tool has been used, including the identification of useful substructures within NIDA networks through activity analysis and through the tracing of causality paths from events to their respective sources. We discuss features of the visualization that allow for the exploration of dense networks and subnetworks. We define analysis goals for the tools, in particular the definition of "similarity" between networks and substructures and the objectives for the recognition of similar substructures. We also introduce a two dimensional visual interface for DANNAs, which includes representation of the physical arrangement of elements on DANNAs, as well as interactions to configure and save the networks. We explore various representations of elements and connections within DANNAs, and we demonstrate the interactions that assist users in evaluating and modifying the networks. Finally, we propose extensions to the tools that will further aid in the exploration and understanding of NIDA and DANNA structure and behavior.

Table of Contents

List of Tables

List of Figures

List of Attachments

Chapter 1

Introduction

In this work, we introduce visualization tools for neuroscience-inspired dynamic architectures (NIDAs) and for Dynamic Adaptive Neural Network Arrays (DANNAs), the hardware implementation of NIDAs. We demonstrate the utility of the visualization tools in exploring and understanding the structure and activity of NIDA networks, as well as in the structure and configuration of DANNA networks. Finally, we describe several extensions to the tools that we believe will further aid in the development and improvement of NIDA and DANNA networks and their associated design methods.

1.1 Neuroscience-Inspired Dynamic Architectures

NIDA networks, representatives of a novel type of artificial neural network introduced by Schuman and Birdwell [26]; Schuman and Birdwell [27]; Schuman et al. [28]; Schuman et al. [29]; and Schuman [30], have been shown to perform well on control, anomaly detection, and classification tasks. Neuroscience-inspired dynamic architecture (NIDA) networks can be viewed as graphs representing the interconnections among two types of components: neurons (nodes) and synapses (edges). Neurons in these networks have two parameters (threshold and refractory period) and exist in a bounded three-dimensional space. They accumulate charge or lose charge from a neutral state and fire when the charge exceeds the threshold; upon firing, neurons enter a refractory period, during which they may still accumulate charge but may not fire, even if the charge exceeds the threshold. Input neurons receive information from the environment, output neurons send information to the environment, and hidden neurons do not interact with the environment. NIDA synapses are directed connections between two neurons and carry charge from one neuron to another. In this sense, the synapses of a NIDA correspond to the axons and synapses in a biological network. Synapses are defined by two parameters: delay and weight. Delay is governed by the length of the synapse (distance between the two neurons the synapse connects) and determines how long it takes for a fire event at the sending neuron of the synapse to affect the charge of the neuron at the receiving end of the synapse. In the current software implementation of NIDA networks, a propagation velocity parameter enforces delay based on synapse length, but other implementations, including the current hardware implementation, could handle delay differently. The weight of the synapse determines how much the synapse charge increases or decreases at the destination neuron.

Unlike many traditional artificial neural networks, the operation of the network is governed by a discrete-event simulation, where event types include fire events in neurons and change in charge events in synapses. One simulated time unit in the discrete-event simulation corresponds to the time it takes for charge to travel one distance unit in the network.

NIDA networks are designed for a particular task using evolutionary optimization. The design process determines the structure of the network (the number and placement of the neurons and synapses), the parameters of the network (such as the thresholds of the neurons and the weights of the synapses), and the dynamics of the network (the delays of the synapses). Advantages and disadvantages of the use of evolutionary optimization to design NIDA networks (and networks in general) are described by Schuman et al. [28]. It is important to note that many of the network structures produced by evolutionary optimization may have equivalent behavior. As a superficial example, the same network rotated or translated in the three-dimensional space will behave exactly the same way as the original network. However, because of the varying parameter values, there are many other structures that are not as easily recognizable as equivalent that may still behave very similarly. The challenge presented by identifying analogous behavior among NIDA network structures is one of the most crucial motivations for designing a tool to explore the networks' behavior.

1.2 Dynamic Adaptive Neural Network Arrays

Dynamic adaptive neural network arrays (DANNAs), introduced by Dean et al. [6], implement the NIDA architecture in hardware. The array uses elements that can be programmed to represent neurons, synapses, or other required elements, and the elements can be rapidly reprogrammed to change the structure of the network. Arrays of up to 85 x 85 elements have been successfully implemented using Field Programmable Gate Arrays (FPGAs). Unlike NIDAs, DANNAs implement delay as a parameter of each element, and the connections between elements are constrained by available ports. Current DANNAs support elements having 8 or 16 ports that can be used for both input and output, and the arrays are scalable to larger quantities of ports in multiples of eight.

DANNA design is based upon and is as intricate as NIDA design, but is further complicated by the constraints of physical element arrays. Visualization tools for DANNA can assist designers in understanding the requirements of both the network and physical array composition. The visualization tools may also interact with the software abstraction layer of DANNA design, allowing users to configure, run, and analyze networks through a graphical user interface. The combination of this interface with visualizations of DANNA networks can provide improved insight into the structure and behavior of the DANNA networks.

1.3 Summary

Some of this work is drawn from the work of Drouhard et al. [7] and Daffron et al. [4]. We will introduce visualization and visual analytics tools to explore the structure and behavior of NIDA and DANNA networks. We present results of utilizing these tools for NIDA networks trained for the task of classification of handwritten digits [18], hand-tooled NIDA networks and substructures, and randomly created DANNA networks. Finally, we discuss future additions to the tool to aid the development of this architecture and design method.

Chapter 2

Background

Data visualization exploits vision and visual working memory to amplify cognition. Since the design of early data plots by William Playfair [32, p. 32], principles of visualization have been developed and refined with the goal of providing maximal insight with minimal graphical complexity. Researchers have studied numerous aspects of vision and perception that have implications for data visualization. In this chapter, we will present findings related to visual thinking, perception and attention, graphical excellence, context and detail, color, quantitative data, pattern recognition, three-dimensional visualization, consistent visual frameworks, and the perception of change. We will also explain the significance of each of these for the visualization of neuroscience-inspired dynamic architecture (NIDA) and dynamic adaptive neural network array (DANNA) networks.

2.1 Visual Thinking

The primary value of data graphics lies not in their aesthetic impact, but in their facilitation of deeper of understanding. To translate observed objects into understanding, graphics can capitalize on one of the most powerful tools for comprehension that humans possess. As Stephen Few explains, "Vision is not only the fastest and most nuanced sensory portal to the world, it is also the one most intimately connected with cognition" [10, p. 29]. The visual cognitive system is complex, but components of it and processes for visual thinking are reasonably well understood.

Components of the visual cognitive system fulfill different roles to assist visual thinking processes. Colin Ware summarizes the units as follows:

- Early Visual Processing: This stage captures low-level features of images and occurs rapidly without need of conscious attention. Early visual processing will be described in greater detail in Section 2.2.

- Pattern Perception: Perceived patterns depend upon the analytical task at hand, but only a small number may be retained in working memory.

- Eye Movements: Eye movements are "scheduled" to explore patterns in order from greatest to least significance. Eye movements between gazes are known as "saccades."

- Intrasaccadic Scanning Loop: During a fixed gaze, information is processed serially at around 40 milliseconds (ms) per item (visually distinguishable shape or region).

- Working Memory: Visual working memory is separate from other forms of working memory, but like other working memory, it can hold only a small number of items at a time. Focused attention controls what is retained in visual working memory and will be described in greater detail in Section 2.2.

- Mental Imagery: Mental imagery is the construction of simple images in the mind. It can be used in conjunction with external imagery for various visual cognitive tasks.

- Epistemic Actions: Epistemic actions are search actions utilized in the exploration of information. For visualization, these actions encompass eye and head movements as well as navigation interactions with media (zoom, pan, etc.) and other interactions with digital graphics.

- Visual Queries: Visual queries seek to validate hypotheses about the data through exploration of visual patterns [36, p. 393].

Each of these components assists in visual thinking algorithms with visual queries forming the "subroutines" of the more complex algorithms. Any of these algorithms may be carried out by computers, but are currently primarily employed by humans. More complex algorithms include: pathfinding on a map or diagram, reasoning with a hybrid of a visual display and mental imagery, design sketching, brushing, small pattern comparisons in a large information space, degree-of-relevance highlighting, generalized fisheye views, multidimensional dynamic queries with scatter plot, and visual monitoring strategies [36, p. 398]. Interactive computer graphics can assist with many of these visual cognitive processes, but any visualization must be well designed to facilitate visual thinking.

Unfortunately, poorly designed data graphics can limit or even impede visual thinking. Edward Tufte laments,

> Much of twentieth-century thinking about statistical graphs has been preoccupied with the question of how some amateurish chart might fool a naive viewer. Other important issues, such as the use of graphics for serious data analysis, were largely ignored. At the core of the preoccupation with deceptive graphics was the assumption that data graphics were mainly devices for showing the obvious to the ignorant [32, p. 53].

When data graphics are not consciously designed to assist visual thinking, they frequently hamper it. According to Few, "Traditional data analysis tools make it unnecessarily difficult to explore data from multiple perspectives, so analysts tend to pursue only a limited set of predetermined questions" [10, p. 104]. Instead of the wide range of visual thinking algorithms described above, poorly designed visualizations restrict thinking.

NIDA and DANNA visualizations have been designed to facilitate exploration and complex visual queries. In particular, they should aid in pathfinding through networks, pattern finding in network structure, identification of key components of networks, and visual monitoring. Several principles described in the following sections contribute to effective facilitation of visual cognition in NIDA and DANNA visualization.

2.2 Perception and Attention

The best data visualizations encourage viewers to study and explore the data. As Christopher Healey describes it, the goal of visualization should be to "build an effective mapping between data values and visual features, so that differences in the features draw the eyes, and more importantly the mind, on their own" [15]. In order to best guide viewers' minds, effective visualization designs must incorporate established principles of perception and attention.

Modern data visualization should take advantage of the knowledge gained from psychological and physiological studies on perceptual capabilities. The human visual system has the capability to process substantially more information from the environment than we can consciously understand. Around 70% of human sense receptors are devoted to vision [10, p. 29]. Data visualization should aim to utilize as much of that sensory data as possible by incorporating known "laws" of perception and attention orientation. Some of the most thoroughly studied of these laws are the gestalt principles originally proposed by 19th century Germany psychologists and philosophers. The primary gestalt laws of perception, as explained by Colin Ware, are:

- Proximity: Objects located or grouped close to each other are perceived as related.

- Similarity: Objects that appear similar are perceived as related.
- Connectedness: Objects linked by lines or other symbols are perceived as related.
- Symmetry: Symmetry between objects or groups of objects can help viewers perceive patterns or make comparisons.
- Closure and common region: Objects enclosed or defined in specific sections are perceived as related.
- Figure and ground: Viewers usually perceive smaller symbols as objects and larger components as landscape or ground behind the objects [36, p. 181].

In order for a visualization to utilize these perceptual principles, it must first capture and hold the viewer's attention. Capturing attention is controlled by older, subcortical visual pathways and is known as "orienting," while holding a viewer's attention over time is called "engaging" and is accomplished by cortical areas linked to the frontal lobe. These two processes can be used to direct attention to interesting components and motivate deeper analysis respectively [15]. When seeking to guide the viewer's attention to key elements of a visualization, it is important to realize that the visual system operates very differently from photography or computer vision. Healey clarifies:

> The goal of human vision is not to create a replica or image of the seen world in our heads. A much better metaphor for vision is that of a dynamic and ongoing construction project, where the products being built are short-lived models of the external world that are specifically designed for the current visually guided tasks of the viewer [15].

Healey's description aligns with Ronald Rensink's coherence theory of visual attention. According to the coherence theory, visual attention is composed of three stages: "early processing" prior to focused attention, focused attention holding objects in a "coherence field" that allows them to maintain continuity through brief interruptions, and releasing attention and the coherence field [24]. Rensink proposes the concept of a triadic architecture for vision composed of systems that are for the most part independent:

1. Early processing: The early processing system is "nonattentional," so without focused attention it allows for the rapid but unstable conceptualization of "proto-objects."

2. Object system: The object system requires focused attention and transforms the conception of proto-objects into meaningful objects.

3. Setting system: A nonattentional system that guides attention based on attributes perceived in early vision [24].

The setting system and early processing are primarily involved with orienting, while the object system is used for engaging. In order to effectively orient and engage viewers, these systems, along with the visual features perceived by each of them, should be considered separately.

The low-level vision system generally processes features that affect the orienting system. According to Healey, orienting to a particular location usually requires a sharp change in luminance, a flicker, or a motion discontinuity [15]. Ware and Few expand on the features related to orienting in their respective descriptions of preattentive processing, which is the perception of data that occurs subconsciously. Features that may be processed preattentively include line orientation, line length, line width, size, curvature, spatial grouping, blur, added marks, numerosity, hue, motion, two-dimensional (2D) spatial position, stereoscopic depth, and convex or concave shape from shading [36, p. 155] [10, p. 39]. Careful encoding of these features will ensure that data visualizations guide viewer's attention to the most important components of information.

Focused attention, required for the engaging system, relies on short-term memory and depends upon cognitive as well as visual factors. Rensink cites evidence of the relationship between visual short-term memory (vSTM) and focused attention [24]. Healey elaborates on the memory requirements between glances, "At most, the details from only three or four objects can be monitored between glances; perception is often limited to only one object at a time. What we see therefore depends critically on which objects in a scene we are looking for and attending to" [15]. In other words, unlike information gleaned through preattentive processing, observations from focused attention are largely dependent upon the viewer's biases and objectives [15]. These findings reinforce the importance of effective encodings for orienting viewers, but they also underscore the need for visualizations designed to facilitate specific analytical tasks.

The primary analytical tasks for which NIDA and DANNA visualizations have been designed are the understanding of structure and behavior of the networks. For both of these tasks, consideration of gestalt principles and preattentive processing have guided the design of the visualization. Feature encodings have been selected with the goal of drawing attention to key relationships between network elements, and feature transitions are used to highlight activity in the networks.

2.3 "Graphical Excellence"

Principles of "graphical excellence" comprise rules of thumb for compelling and effective graphical representations of data. Tufte defines graphical excellence as, "that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space and graphical excellence requires telling the truth about the data" [32, p. 51]. He enumerates the following principles for graphical displays:

- Represent the data.
- Cause viewer to think about the substance rather than the methodology.
- Avoid distortions.

- Condense quantitative information.
- Make large data sets coherent.
- Encourage the eye to make comparisons between data.
- Reveal several levels of detail.
- Serve a reasonably clear purpose.
- Integrate closely with statistical and verbal descriptions of a data set [32, p. 13].

The primary aim of graphical excellence, supported by each of these principles, is to showcase the data with as few distractions as possible.

Clear visualizations must also distinguish appropriately between data features and avoid distorting the data. Tufte proposes that the number of visual feature dimensions should not exceed data dimensions [32, p. 71]. Ware advises, "Use different visual channels to display aspects of data so that they are visually distinct" [36, p. 145]. Appropriate encodings of visual features for data features are critical to understanding of data graphics.

Visualization researchers agree that another key component to high quality data visualization is conciseness. Ware emphasizes that visual queries are most efficient for compact visual displays that maximize the amount of information that can be perceived in a single glance [36, p. 141]. Tufte describes graphical succinctness in terms of "data density" and "data ink ratio." He argues that effective displays maximize both data density, which is the number of entries in a data matrix divided by area of data display [32, p. 161] and data ink ratio, the ratio of ink that communicates essential information to ink that does not communicate valuable information [32, p. 93]. Tufte's principles extend Ware's by requiring the removal of unnecessary decoration, which he calls "chartjunk," in addition to the compact representation of the data itself. He even advocates the removal of some traditional graphic elements, stating, "One of the more sedate graphical elements, the grid, should usually be muted or completely suppressed so that its presence is only implicit lest it compete with the data" [32, p. 112]. Essentially, the ideal data visualization shows a compact representation of the data and not much else. In fact, Tufte's principles of graphical revision and editing enforce this concept explicitly:

1. "Above all else show the data.
2. Maximize the data-ink ratio.
3. Erase non-data ink.
4. Erase redundant data-ink.
5. Revise and edit" [32, p. 105].

The principles for graphical excellence are not universally applicable, but they should certainly be taken into consideration in the creation of data visualizations. In our design of NIDA and DANNA visualizations, we have sought to adhere to these principles in most cases. We have attempted to eliminate distracting elements such as grids and maximize the data ink ratio, and we have used different visual channels to encode distinct data features. In some cases, we have used combinations of visual features, such as color and shape, to encode data features. We have also used conjunctions of features including size and hue variation to highlight activity. Although these design decisions may violate the principle of limiting visual feature dimensionality, we believe that in this context they clarify rather than distort the data.

2.4 Context and Details

Beyond the foundational principles of graphical excellence, combinations of context and detail views in visualizations enhance users' understanding and ability to explore data. Tufte asserts that there are a minimum of three depths to visualizations, which he describes as, “1) what is seen from a distance, an overall structure usually aggregated from an underlying microstructure; 2) what is seen up close and in detail, the fine structure of the data; and 3) what is seen implicitly, underlying the graphic that which is behind the graphic” [32, p. 155]. Ben Schneiderman encapsulates users’ needs for different depths in a pithy mantra, “Overview first, zoom and filter, then details-on-demand” [10, p. 84]. Rensink argues that it is impossible for a visualization to simultaneously display context and detail [24], but he likely means only that a single view is not capable of representing every layer of detail about the data. Instead, it will be necessary for visualizations to support interactions that users can apply to see more or less detail on demand.

Filtering, aggregation, zoom/pan/rotate, and highlighting are among the most important interactions that can help users navigate different levels of detail. Few explains that filtering helps remove distractions to streamline visual queries, while aggregation can provide context at multiple levels [10, p. 64, 69]. Other interactions allow for focus on key pieces of data within a single level of detail. Zoom and pan—or rotate for three-dimensional (3D) environments—should be supported as mouse interactions, according to Few [10, p. 76]. Highlighting is another key technique for drawing attention to some components within a larger context. To highlight important components of data graphics, Ware suggests either reducing contrast for less important regions or “haloing,” or increasing the local background luminance contrast for highlighted regions [36, p. 78]. These techniques allows users to assimilate different levels of detail about the data and isolate focus on regions of interest.

The visualizations for NIDA and DANNA incorporate several of these techniques to provide users both context and details-on-demand. In the current implementation, aggregation of network data is not supported, but some filtering is supported to isolate sub-structures of interest. Highlighting is used to draw attention to active areas of NIDA networks and user-selected elements in DANNAs. Zoom is supported for both tools, and rotate or pan interactions are implemented for NIDA and DANNA visualizations respectively. Additional interactions that would be useful to provide context and/or detail are described in Chapter 7.

2.5 Color

Color is an ideal visual feature for distinguishing between categorical values. Hue is preattentively processed by the low-level vision system, so it does not require focused attention to discriminate between different categories designated by hue. Moreover, families of categories may be identified using different levels of saturation and lightness for the same hue, though only around two per family are perceptibly distinct [36, p. 128]. Dual encodings using color and shape are another useful technique for differentiating types of symbols [36, p. 151]. However, only a modest number of colors should be used for effective encodings. As Post and Greene demonstrated in 1986, no more than around nine colors including white are consistently distinguished [36, p. 110]. Based on these and other similar findings, color has been identified as an ideal visual marker for category provided that only a small quantity of types need be identified.

NIDA and DANNA visualizations make use of the known perceptual principles related to color. Hue and shape redundantly encode type of element, and hue differentiates category within element type (e.g., input, hidden, or output neurons). A small number of colors is used in each color scheme so that differentiating between colors requires minimal cognitive effort. Adjusted saturation and lightness for color families are proposed in Chapter 7 to encode neuron charge level, but no more than three levels of charge would be tracked using this scheme.

2.6 Visualization of Quantitative Data

Understanding quantitative values is often key to comprehending data visualizations, so quantitative encodings should be chosen with care. Few underlines the value of quantitative representations, saying,

> When we represent quantitative information in visual form, our ability to think about it is dramatically enhanced. Visual representations not only make the patterns, trends, and exceptions in numbers visible and understandable, they also extend the capacity of our memory, making available in front of our eyes what we couldn't otherwise hold all at once in our minds [10, p. 6].

To allow for this amplification of visual thinking, quantitative values must be encoded using visual features that can be processed quickly and interpreted accurately. Several preattentively processed features lend themselves naturally to quantitative comparisons. Among these are size, lightness/darkness (on contrasting backgrounds), vividness, 2D position on display, length/width, intensity, and blur [36, p. 168] [10, p. 41]. Of these, width, size, intensity, and blur may be less precisely interpreted. Viewers perceive these features as inherently quantitative in data visualizations. Other encodings may be used, but they will likely be perceived as arbitrary, and may require focused attention and more time to process.

In addition to careful selection of visual features, proportional representation and some specialized techniques improve the comprehension of quantitative data visualizations. Tufte asserts that the representation of numbers should be directly proportional to data values in quantitative graphics [32, p. 56]. Adhering to this rule helps avoid distortion of the data, which is required for graphical excellence. To encode multiple quantitative values for each data point, Few suggests heat maps and a special derivation thereof, tree maps [10, p. 45]. These maps utilize two preattentively processed features, 2D location and intensity (or vividness), allowing for rapid processing of two quantitative values for each data item. Proportional representation and tree map techniques should be used to improve comprehension of quantitative visualizations.

Several components of the NIDA and DANNA visualizations are quantitative in nature, and we have incorporated these principles into their design. Synapse length represents delay in the networks, and synapse and neuron locations are depicted to scale. Stroke weight encodes the magnitude of synapse weight with direct proportionality in the NIDA visualization. We do not expect that viewers will be able to distinguish between small differences in stroke weight, since width is generally not perceived precisely. However, the use of proportional representation allows viewers to compare significant differences between synapse weights. In maps of causality subnetworks presented in Chapter 5, size is used to encode frequency of occurrence. As with synapse stroke weights, we have used this feature to facilitate visual comparisons, but we do not expect viewers to distinguish precise frequencies using this encoding. We have also used numerosity and 2D location of glyphs to represent the port level and number in the DANNA grid view presented in Chapter 6. Other proposed quantitative visualizations for NIDA and DANNA networks are described in Chapter 7.

2.7 Pattern Recognition

The choice of encoding and visual grammar has a significant impact on pattern finding capabilities of the viewer of a data visualization. If a visualization needs to represent multiple data dimensions, Ware advises selecting encodings that rely on separate visual channels for separate attributes. For example, motion or spatial grouping might represent one feature, while color or shape encodes another [36, p. 161]. Relying on separate visual channels ensures that these features can still be processed preattentively. To avoid visual interference, Healey advises, "The most important attributes (as defined by the viewer) should be displayed using the most salient features. Secondary data should never be visualized in a way that masks the information a viewer wants to see" [15]. These guidelines help promote visual pattern recognition in data visualizations, and they contribute to overall graphical excellence.

Pattern finding is also facilitated through coordinated views and recognizable visual grammars. Coordinated views are different visual representations of the same data that may highlight different attributes of the same data. If coordinated views are linked and use brushing, then the selection or highlighting of certain data points in one view highlights the same data points in the linked views. Few outlines the benefits of brushing and coordinate views, stating,

> The ability to see data from multiple perspectives simultaneously brings a great deal of information together, reducing our need to rely on limited working memory. When these different views of the data are tightly coupled, the potential of faceted analytical displays can be expanded dramatically, especially through filtering and brushing [10, p. 112].

As he indicates, not only do these coordinated views bring together context and details, they also mitigate the limits of visual working memory. The conjunction of these factors expands the pattern-finding capabilities of the visual thinking system. Pattern-finding capabilities may also be enhanced by the use of known "visual grammars," or abstractions for particular types of data that are so commonly used they are instantly recognizable. For graphs (node-link diagrams), the visual grammar is composed of small symbols or enclosed shapes for entities and lines to represent relationships or paths between them [36, p. 222]. Use of a visual grammar such as this reduces the cognitive load required to understand a data visualization. When data visualizations use visual grammars in conjunction with coordinated views, pattern recognition is a much simpler task.

NIDA and DANNA visualizations are designed to help users understand the structure of the networks and recognize patterns within them. In order to assist users in these tasks, the visualizations use color and shape to differentiate entity types, spatial grouping and the grammar of node-link diagrams to encode structure and relationships, and motion and highlighting to represent activity and user interactions. We have designed the visualizations to encode key data attributes using different visual channels and features that can be preattentively processed so that patterns are more easily recognizable. In Chapter 7, we propose coordinating views and brushing to further boost pattern finding capabilities.

2.8 Three-dimensional Visualization

Three-dimensional (3D) data visualization should be used only when it best supports necessary analytical tasks for a specific data set. 3D visualization involves complicated implementation details, but more importantly, it is not the best means to assist in pattern finding for many tasks. According to Ware, the brain is generally best suited to identify patterns in 2D [36, p. 239]. Nevertheless, 3D visualizations are appropriate for some tasks and may be the primary means of data visualization in the future. Ware affirms, "The strongest argument for the ultimate ascendancy of 3D visualization systems, and 3D user interfaces in general, must be that we live in a 3D world and our brains have evolved to recognize and interact within 3D" [36, p. 290]. In a 3D world with high dimensional data, 3D visualization systems can allow for efficient data exploration when implemented carefully.

Experiments have demonstrated that complex 3D graphs are best understood through 3D visualizations as long as depth cues and viewport control are implemented appropriately. Depth cues can be achieved through linear perspective, texture gradient, size gradient, occlusion, depth of focus, shape-from-shading, vertical position, relative size to familiar objects, and cast shadows [36, p. 240]. However, motion parallax is more powerful than any of these cues in enabling perception of more information [36, p. 290]. Ware and Franck showed in a 1996 experiment that superior path tracing ability for large 3D graphs was achieved with 3D visualizations using stereoscopic and motion depth cues [36, p. 275]. In order to effectively interact with 3D visualizations, one of the following metaphors should be used:

1. World-in-hand: The user metaphorically rotates or moves the entire world closer.

2. Eyeball-in-hand: The user metaphorically manipulates a camera's direction and location in the 3D space.

3. Walking: The user navigates the environment by "walking" through it.

4. Flying: Smoother than walking, the user controls velocity and up, down, backward, and forward movement with hand motions [36, p. 356].

With these interactions and strong depth cues, 3D environments can enhance users' pattern finding abilities.

The NIDA visualization utilizes a 3D environment to help users understand structure and activity within NIDA networks. Linear perspective, relative size, occlusion, and motion parallax are utilized as depth cues. The user may zoom and rotate the viewport according to the eyeball-in-hand metaphor described above. Although the complexity of the 3D environment has limited the interactions available to users, we believe that the benefits of 3D space for understanding these complex networks outweigh the limitations. We have outline interactions that may be implemented in the future in Chapter 7.

2.9 Consistent Visual Framework

The maintenance of a consistent visual framework in data visualizations prevents data distortion and helps provide users with the combination of context and details. Ware recommends the preservation of visual mappings of data be across various views and the use of devices such as frames and landmarks to maintain visual continuity [36, p. 341]. He also asserts that a consistent visual framework in a 3D environment requires persistence of a sufficient number of objects from frame to frame to judge position [36, p. 355]. Use of these techniques helps viewers maintain a sense of the full context of data space when zooming and filtering data for details.

Both NIDA and DANNA visualizations are designed to maintain a consistent visual framework. The same visual mappings are used for most views (with the notable exception of causality subnetworks described in Chapter 5), zooming and filtering are limited to ensure persistence of some objects across frames. We made these design decisions for visual continuity with the aim of supporting visual queries and pattern recognition.

2.10 Perception of Change

Change blindness is the phenomenon of an observer's failure to notice changes in visual stimuli even when they are large and not at the periphery of the field of view. To mitigate the risk of change blindness, the factors that influence it should be taken into account in the design of data visualizations. As Healey describes it, "Change blindness is not a failure to see because of limited visual acuity; rather, it is a failure based on inappropriate attentional guidance" [15]. Rensink's experiments have shown that focused attention is needed to perceive change, and objects considered interesting are less likely to be affected by change blindness. Rensink also reports the findings of memory studies indicating that only around four items can be monitored in visual short-term memory, so layout changes that affect more than a few objects are most susceptible to change blindness [24]. Given these findings, we can best prevent change blindness by guiding viewers' attention to a small number of "interesting" objects in a visualization.

The manner in which changes affect the ground (not the objects) of a visualization also impacts the degree of change blindness that occurs. Experiments performed by Wang and Simons, as well as later experiments by Rensink, have shown that change detection depends upon spatiotemporal continuity in the representation of data. Some experiments showed that observers can better detect changes in a rotated layout if the observers, rather than the layout are rotated [24]. For saccade-contingent changes, Rensink observes, “In all cases, observers are generally poor at predicting change. Indeed, this is true for position change if even only one item is present, provided it has no global frame of reference” [24]. In other words, a consistent visual framework and specific types of layout interactions can improve observers’ ability to detect change.

In designing NIDA and DANNA visualizations, we have attempted to mitigate the effects of change blindness to the extent possible. As explained in Section 2.2, the designs seek to guide users’ attention to key components of the visualization and enhance their perception of change using highlighting techniques described in Section 2.4. We have also tried to maintain visual continuity through a consistent visual framework, as detailed in Section 2.9, to avoid confounding the detection of change. With these design features, we anticipate that viewers will maintain sufficient context in the representation of data to observe significant changes—primarily activity in the networks.

2.11 Summary

This chapter provided a brief introduction to key components of the visual and visual thinking systems, including: visual thinking, perception and attention, graphical excellence, context and detail, color, quantitative data, pattern recognition, three-dimensional visualization, consistent visual frameworks, and the perception of change. The applicability of each of these topics to NIDA and DANNA visualizations has also been addressed. The findings that we consider most relevant to our visualization work on NIDA and DANNA networks are the following:

1. Vision and cognition are profoundly related (Section 2.1).
2. Visual thinking algorithms that should be supported for NIDA and DANNA networks include: pathfinding, pattern recognition in network structure, identification of key components of networks, and visual monitoring (Section 2.1).

3. The gestalt principles of closure and common region, proximity, similarity, connectedness, and symmetry are particularly relevant for NIDA and DANNA visualizations (Section 2.2).

4. The following features can be processed preattentively (without focused attention): line orientation, line length, line width, size, curvature, spatial grouping, blur, added marks, numerosity, hue, motion, 2D spatial position, stereoscopic depth, and convex or concave shape from shading (Section 2.2).

5. Since visual short term memory only allows for the monitoring of around four objects between glances, it is important to engage a viewer's focused attention on items of interest and ensure that they can be effectively observed in a single glance. (Section 2.2).

6. Graphical excellence requires an accurate representation of the data and the avoidance of distractions and distortions (Section 2.3).

7. Data visualizations should allow viewers to explore various levels of detail, or as Ben Schneiderman puts it, "Overview first, zoom and filter, then details-on-demand" (Section 2.4).

8. Only around nine color hues can be reliably distinguished (Section 2.5).

9. Families of colors can be identified by varying saturation and lightness for the same hue, but only around two colors per family are distinguishable (Section 2.5).

10. The best graphical features for quantitative comparisons are lightness/darkness, vividness, 2D position, size, length/width, intensity, and blur, but the last four may be perceived less precisely (Section 2.6).

11. To facilitate comprehension of the data, the graphical representation of quantitative values should be proportional to the data values (Section 2.6).

12. Patterns are more reliably recognized when the perception of features used to represent different data dimensions relies on different visual channels (Section 2.7).

13. Recognizable visual grammars also improve pattern finding abilities (Section 2.7).

14. 3D visualizations improve understanding of complex 3D graphs when depth cues and viewport controls are appropriate (Section 2.8).

15. Motion parallax is the most powerful depth cue (Section 2.8).

16. Visualizations should maintain a consistent visual framework to provide visual continuity and prevent distortions of the data (Section 2.9).

17. Change blindness—the failure to notice even large changes—can be mitigated with appropriate guidance of viewers' attention (Section 2.10).

The next chapter provides background on NIDA and DANNA networks, as well as on visualization techniques that have been applied to neural networks.

Chapter 3

Related Work

3.1 NIDAs and DANNAs

Fields related to neuroscience-inspired architecture are rapidly growing in computer science and engineering. Previous works have introduced a neuroscience-inspired dynamic architecture (NIDA) and associated design method and have demonstrated the utility of NIDA networks on several problems, including anomaly detection [27], control [26], and classification [29]. This new architecture makes use of dynamic properties and distributed memory to address these varying problem types. However, understanding the behavior of this new network type can be a daunting task, especially without prior knowledge as to what to expect. Developing an intuition about how a network behaves is an important factor in determining the utility of that network for tasks, as well as how to exploit features of the network in the learning/design process. With this in mind, we believe a visual analytics tool tailored specifically for NIDA networks can be an extremely useful aid to better understand the types of structures and behaviors that are produced by the design process.

Several efforts are documented in the literature that explore novel computational approaches motivated by neuroscience. We restrict our attention here to those that implement spiking, or event-driven, behaviors. The Blue Brain project [20], the Human Brain Project [21] and IBM's cognitive computing project [22] attempt to model what occurs in biological brains with high fidelity, whereas hierarchical temporal memory (HTM) [14] takes inspiration from biology, but does not restrict the operation and training of HTM networks to what occurs in biological brains. Other machine learning techniques, such as deep learning [1] also take some of their inspiration from neuroscience and theories of learning in the brain. While NIDA methods take inspiration from biological systems, they do not attempt to accurately model or represent biological processes. Rather, we are motivated by neuroscience but focused upon development of effective computational architectures. Dynamic Adaptive Neural Network Array (DANNA) hardware implementations of the NIDA approach have also been explored by Dean et al. [6]. DANNA represents a class of neuromorphic computing architectures. NIDA network design is based on evolutionary optimization, a method which has been used in the literature to design traditional artificial neural networks [38, 12].

3.2 Visualization Methods for Neural Networks

Numerous visualization methods have been developed to help better understand the learning processes and behaviors of artificial neural networks. Darrah [5] presents a survey of techniques intended to help users make sense of the interactions between elements in neural networks and the ways that those interactions evolve over time. Among other techniques for verification and validation, Darrah introduces Hinton diagrams, Bond diagrams, hyperplane diagrams and animators, Self-organizing Maps (SOMs), and Voronoi diagrams.

Hinton diagrams [16] were among the first means of visualizing structural relationships within neural network elements. Compact graphs for each neuron in a network N are arranged in the same order as the nodes that they represent. The graph for a given node n contains small squares for each of n's edges in N, also in an arrangement that is spatially representative of the network. Black-colored squares can be used to indicate negative weight edges, while white squares stand for positive weight edges. The size of each square represents the magnitude of the edge weight, enabling the user to quickly evaluate which nodes in N are likely to have the most influence on n's behavior.

Several other ANN visualization techniques are related to or based in part on Hinton Diagram concepts. Weight Visualization diagrams and curves (WV-diagrams and WV-curves), described by Bischof et al. [2] and Bischoff et al. [3], respectively, show the same spatial information that Hinton diagrams show in different ways. WV-diagrams were designed to show the same values as Hinton diagrams using less space, so they use a grayscale value instead of size to encode weight and require a single pixel for each weight value represented. Color values may be used instead of grayscale to allow for better perception of weight distinctions. When numerical values are represented in a distributed manner, WV-curves can show weights more intuitively. WV-curves encode weight using distance in a coordinate plane. This method facilitates the perception of patterns across multiple input channels.

A more recent visualization mechanism derived from Hinton diagrams is known as a connection map. Connection maps are similar to WV-diagrams in that they use pixel values to encode the strength of relationship between elements, but they have been used by Thiessen to visualize more general graph problems using Pearson's bivariate correlation as a measure of similarity [31]. Connection maps are generated by first assigning each node of a k node graph to a position within a grid containing k locations. Then the similarities of a node n to all other nodes are calculated and represented as grayscale or color values within pixels assigned to those respective nodes in n's grid. Connection maps have been shown in a user study to have intuitive meaning and to facilitate understanding significantly better than visualizations of the graphs themselves [31].

An alternative and more scalable tool for pattern recognition weights is the Quadrant-Distance (QD) graph introduced by Linnell [19]. QD graphs can show the weight vectors of the network at any point during training and are capable of revealing patterns in networks of various architectures and sizes. QD graphs map a set of network weights to vectors in N-dimensional space where the number of dimensions are defined in Equation 3.1.

$$\begin{aligned} N = (number\_of\_input\_nodes + 1) * (number\_of\_hidden\_nodes) \\ (number\_of\_hidden\_nodes + 1) * (number\_of\_output\_nodes) \end{aligned} \tag{3.1}$$

QD graphs then use simple metrics, such as Euclidean distance from the origin or the number of the quadrant in which the weight vector lies, in order to visualize the weight space using simple line graphs [19]. The significance of the patterns exposed by QD graphs is not always intuitive, but in conjunction with other knowledge of the networks, the graphs have demonstrated utility for the analysis of neural networks. QD graphs are particularly useful in comparisons between networks with different initial weights and in the tracing of network learning during training.

Other researchers have developed alternative methods for further exploration of the input space, structure, learning, and accuracy of neural networks. Graph visualizations [37], scatterograms [8], dual-space interactive weight visualizations [33], and three dimensional simulations [40] are some of the techniques that have been proposed.

3.3 Self-Organizing Maps (Kohonen Maps)

Self-Organizing Maps (SOMs), introduced by Kohonen et al. [17], are among the most widely used visualization techniques for neural networks. They have frequently been used to reduce dimensionality of the input space and cluster input data so that tasks required of the networks can be more easily understood [25, 35, 39, 11].

SOMs have also been used by Uzak et al. [34] to simplify the visualization of network behavior in clustered response-funtion plots. Response-function plots show the response of individual neurons to various inputs. Since visualization of very large networks can be limited by human abilities to follow changes in large numbers of pictures, these authors propose visualizing only the representatives of clusters of neurons with "similar responses." Clustered response-function plots thus reduce visualization requirements and visual clutter by a factor of the cluster size.

3.4 Neuroscience Simulation Visualizations

Although not directly applicable to NIDA visualization, neuroscience simulations include useful concepts for the analysis of neural network topology and behavior. Two neuroscience simulations with powerful three dimensional visualizations are NeuGen [9] and neuroConstruct [13]. NeuGen was designed for the realistic simulation of biological neural networks, and it provides interactive visualizations of complex networks that allow highlighting of specified network regions [9]. neuroConstruct, also designed for more anatomically realistic network visualizations, boasts additional features. It has settings for adjustable transparency to assist users in exploring different layers of the network, as well as adjustable levels of detail for viewing simulated network activity. neuroConstruct users may also interactively highlight connections of specified cells [13]. Both tools are freely available to the public.

3.5 Summary

This chapter introduced Neuroscience-Inspired Dynamic Architecture (NIDA) and Dynamic Adaptive Neural Network Array (DANNA) networks, including their structure and behavior, as well as some of the problems on which these networks have shown a strong performance. Also presented in this chapter were several of the visualization tools and techniques that have been used to explore and analyze neural network behavior, such as Hinton diagrams, Self-Organizing Maps, and neuroscience simulations. The next chapter provides a brief overview of the research contributions of the NIDA and DANNA visualization tools produced for this work.

Chapter 4

Overview of Results

This chapter outlines the primary research contributions of this work. We will present the basic features available in the NIDA visualization tool and DANNA visual interface, and we will discuss the analysis tasks for which these tools have proven useful. Links to relevant sections in other chapters are provided for additional details about results.

4.1 Three Dimensional NIDA Visualization Tool

The three dimensional NIDA visualization tool allows users to explore the complexities of NIDA network structure and observe patterns in activity. Users may interactively explore networks in the environment or define preset interactions to be rendered in high frame-rate videos. Analytics features, such as causality path tracing, assist users in identifying useful substructures that may help improve the NIDA design method. An example image from the NIDA visualization tool is shown in Figure 4.1.

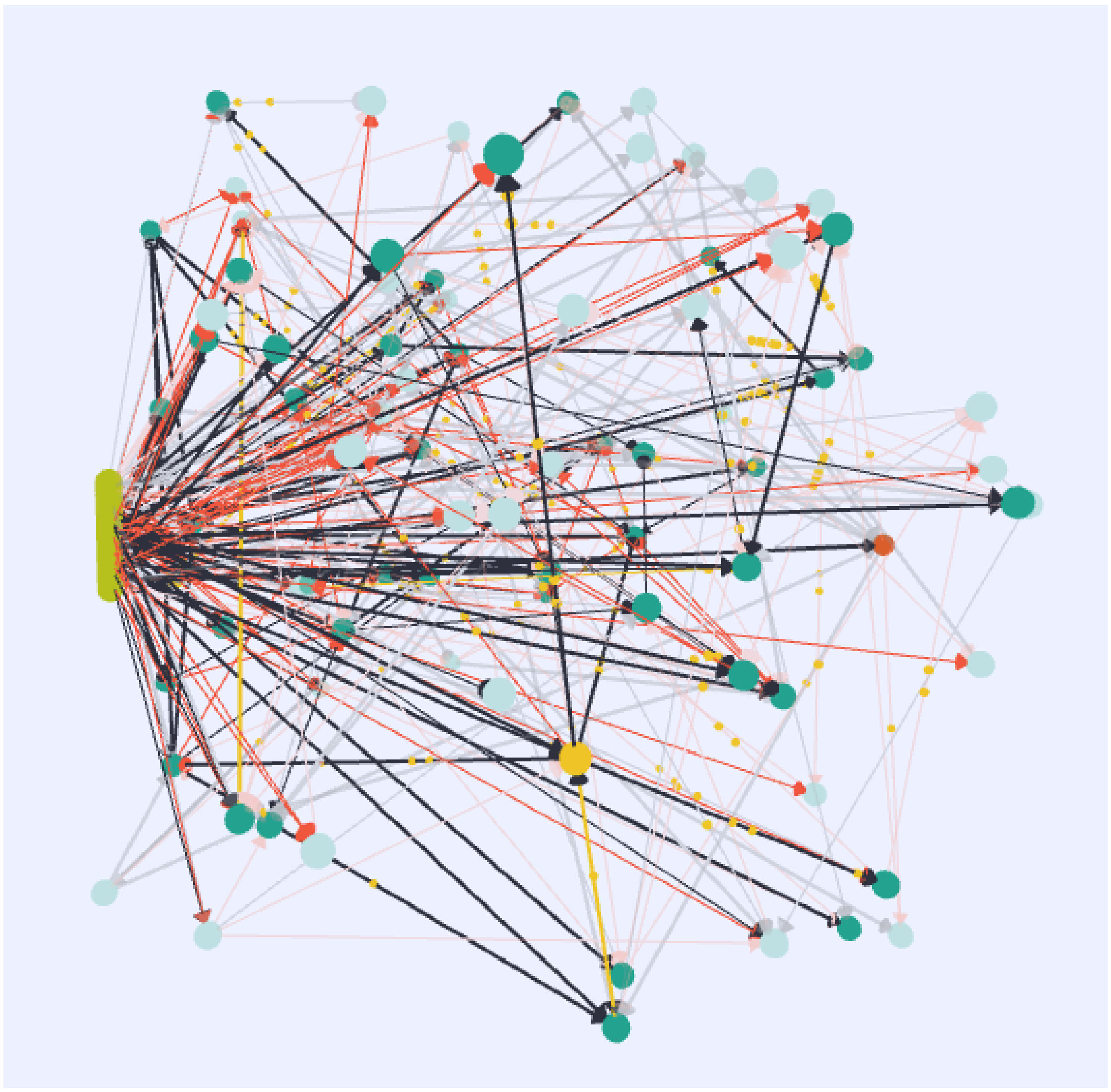


Figure 4.1: An example image created by the three dimensional NIDA visualization tool showing a network trained to recognize the handwritten digit 7 during the processing of an input image of the digit 1. Multiple visibility modes are shown, as well as highlighting of active elements.

4.1.1 Exploration

The NIDA visualization tool has helped us explore the learning process of NIDA networks. We have used it to analyze whether propagation of events through the network occurs as expected. We can use the tool to examine the various components of the network, to determine their activity levels, and to identify the time periods within the simulation time frame for which neurons, synapses, and substructures are most active. Interactive mode may be used to explore networks and generate hypotheses, and video mode with predefined interactions is particularly useful for comparing networks or the behavior of a single network on different inputs. (See Section 5.2.1.)

4.1.2 Highlighting and Filtering

We recognize the value of identifying events that will or should affect the firing of neurons. Longer synapses (synapses with greater delay) sometimes prevent expected firings or facilitate unexpected firings. We also stipulate that analysis of inhibitory synapses and their effect on the firings of neurons will be useful for the understanding of NIDA networks. The highlighting of active elements through increasing size and contrasting colors effectively draws attention to events and locations of interest within the network. In order to allow exploration of subnetworks of elements that are active over multiple inputs to the same network, filtering features are provided. Currently, filtering is implemented through the identification of useful substructures and rendering only elements contained within those substructures. (See Section 5.2.2.)

4.1.3 Scaling

Given the varying densities and granularities of NIDA networks, we have implemented adjustable scaling to facilitate exploration of the networks at different levels of detail. As an example, affective systems, described in detail by Schuman and Birdwell [26], are sometimes implemented as smaller networks within networks of larger scale. In one approach, the granularity (minimum distance between neurons) of an affective system is an order of magnitude smaller than the granularity of the network within which it exists. Scaling features within the visual analytics tool allow users to explore both the larger network and its affective system(s). (See Section 5.2.4.)

4.1.4 Similarity between Networks

We define different types of similarity based on the behavior of one or more networks over multiple input data. "Similar behavior" must be defined by NIDA developers, but we hypothesize that similar networks will have analogous output patterns given the same inputs, and we also expect that similar substructures may exist in different networks that behave similarly. One of our objectives is to facilitate the identification of substructures that exist in distinct networks, particularly networks trained for the same task. As a precursor to this task, we have used the visualization tool to identify some useful substructures within individual networks whose behavior we may wish to emulate in networks trained for the same task. We have examined how the substructures interact with input and output neurons. (See Section 5.4.2.)

4.1.5 Identification of Useful Networks/Substructures

We have established two techniques to assist NIDA developers and designers in identifying useful substructures: activity-based identification and causality paths. The activity-based method requires only an analysis of the most active neurons and synapses within the networks. However, the results of extending the NIDA evolutionary optimization method to incorporate this technique have not been promising, so we believe the strategy may be too simplistic to identify useful substructures. We anticipate that our causality-based method will target more substructures of interest. Causality paths can trace an arbitrary event e in a simulation back through all precipitating events to the initiating input event(s) causing e to occur. They may also trace e forward through the simulation to show all events triggered by e. We provide static views of these causality paths as well as animated traces forward through time, restricting the view of the network to only the elements along the causality path. We also provide functionality for "causality subnetworks," which allow the selection of multiple events whose causality can be traced. (See Sections 5.4.4 and 5.5.)

4.2 Two Dimensional DANNA Visual Interface

In addition to the three dimensional NIDA visualization developed for this work, we have produced a two dimensional interactive visual interface for DANNA. This tool has been designed for DANNA developers, but it is also intended to have sufficiently intuitive interactions such that researchers first beginning their exploration of DANNA networks can gain insight into the features of the networks and efficiently explore their behavior. The default view of the DANNA visual interface is shown in Figure 4.2.

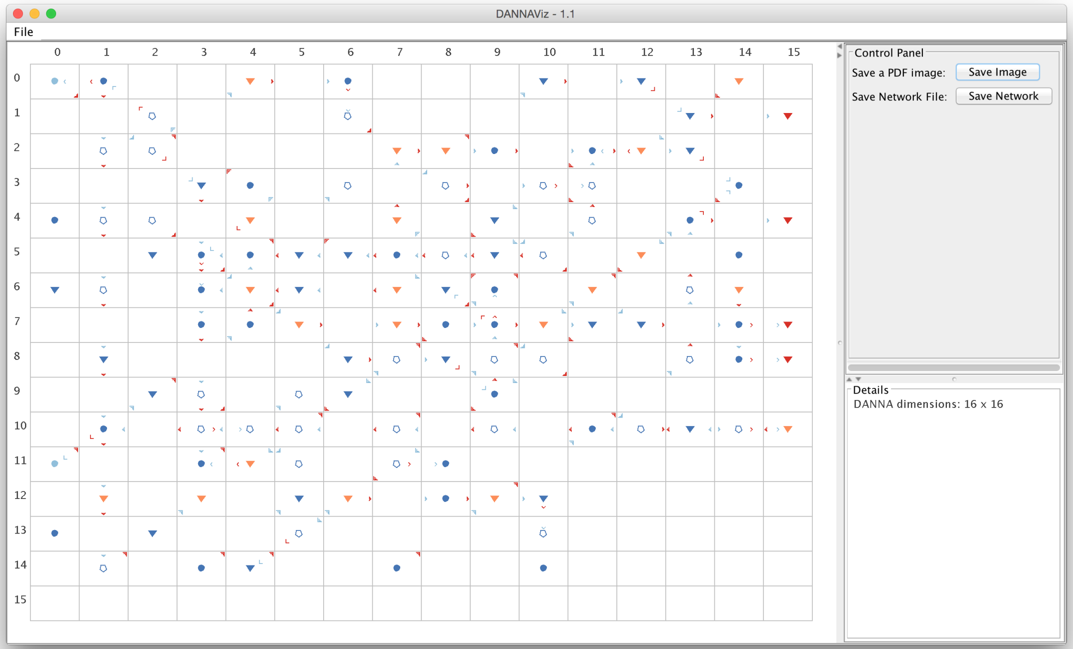


Figure 4.2: The default view of the DANNA visual interface showing a network loaded onto a 16 x 16 DANNA.

4.2.1 8-connection Visual Interface

In the initial implementation of DANNA, each element could connect to its 8 immediate neighbors within the DANNA grid. We implemented a grid view for this connection scheme that represents neurons as ellipses and synapses as arrows from the pre-synaptic neurons to the post-synaptic neurons. The interface features interactions that allow users to hover over and highlight elements to view additional information, modify elements using a configuration menu, and save resulting images or network files. (See Section 6.2.)

4.2.2 16-connection Elements

The updated DANNA implementation allows each element to connect with its 8 neighbors from two hops away, as well as its 8 immediate neighbors. This implementation also includes "passthru" elements that may connect neurons and synapses or multiple synapses in order to facilitate greater fan-out of elements within the array. Since other elements may be included in the future, we have updated the grid view to encode elements with different shapes and to distinguish node type (input/hidden/output) using color. The same interactions with elements are available as in the 8-connection visual interface. Additionally, we have added zoom and pan features to offer users greater freedom in exploring DANNAs. (See Sections 6.3.1 and 6.3.2.)

4.2.3 16-connection Connections

Larger DANNAs may require more than 16 connections per elements, so we have modified the visual interface to represent connections at multiple levels and to be automatically extensible to greater numbers of connections. Connections are represented in rings around the element shapes within the DANNA grid, mirroring the physical arrangement of connections in the DANNA hardware. Arrow-like symbols indicate the direction and type (input/output) of a connection, and the number of marks per symbol encodes the level of the connection. These encodings utilize space efficiently within the grid view, so relationships between elements are visible even within large networks. (See Section 6.3.3.)

4.3 Summary

In this chapter, we have enumerated the fundamental contributions of this work. In the following two chapters, we will describe the visualizations for NIDA and DANNA networks in detail.

Chapter 5

Three Dimensional NIDA Visualization

5.1 Background on NIDA Networks Visualized

Most of the NIDA networks appearing in this chapter were designed by Catherine Schuman as part of an ensemble method used in the classification of handwritten digits from the MNIST data set [18]. Each network is designed (using evolutionary optimization) to identify a particular digit d by firing its output neuron in a pre-defined time window (the last 50 time steps of a 500 time step simulation; other timings can be used) if the image is of the digit d. The network should not fire in the final time window if the image is of a digit other than d. The fitness function is described in more detail by Schuman et al. [29]. In one implementation, each network has 28 input neurons and one output neuron, and each network in the ensemble receives each image as input. Each image of a handwritten digit is 28 by 28 pixels. The networks "scan" the image, receiving one row or one column at a time. If the output neuron of a network associated with digit d fires during the last 50 times steps of simulation, that network "casts a vote" for digit d for that image. The digit receiving the most votes is the decision of the ensemble (Figure 5.1). The evolutionary optimization produces a single network trained to recognize a particular digit on a set of training images. Based on the performance of all resulting networks on the training images, the top ranking networks for each digit are assembled to produce the ensemble. An ensemble of 2600 networks produced a classification accuracy of 90.6 percent on the testing set of images (images that were not used during training) [29, 30].

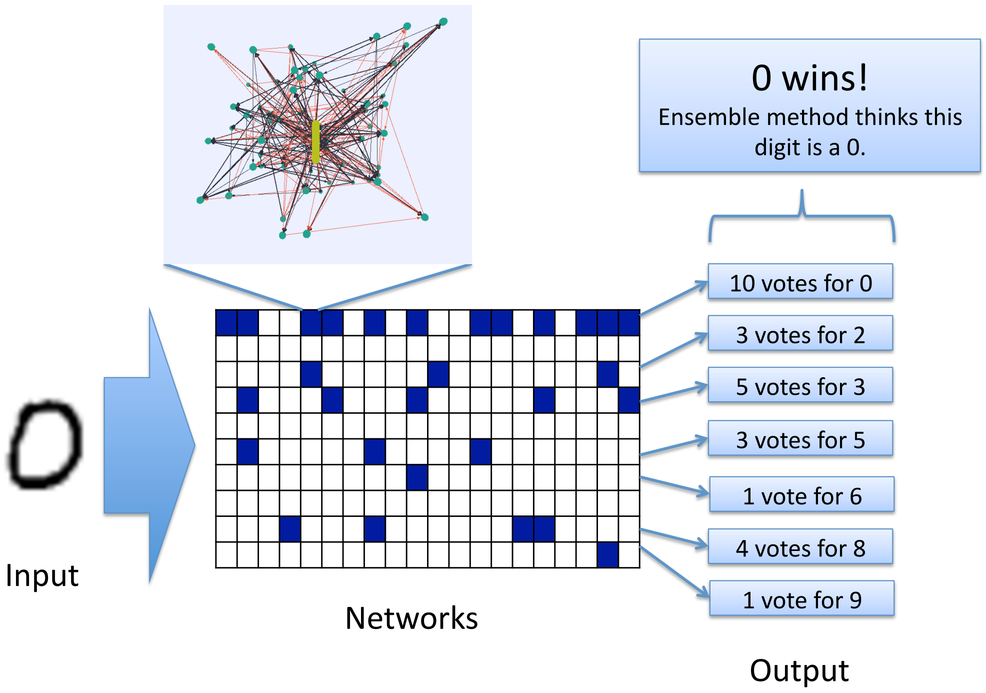


Figure 5.1: A voting scheme amongst networks of an ensemble is used to determine the digit for a particular image. Each network in the ensemble (represented by a square on the grid) receives an image on input and simulates activity within the network. Based on that activity, the network may or may not cast a vote (casting a vote is represented by shading in the grid). The digit with the most votes is the guessed digit for the ensemble. [29, 7]. (Figure created in part by Catherine Schuman and used with her permission.)

5.2 Primary Features

5.2.1 Modes of Operation and Feature Encoding

In order to examine the behavior of the NIDA networks, we created a three dimensional (3D) network model using Processing [23] to represent the structure of a given network to scale. Visualization of spatial information is particularly important for NIDA networks, since their structures are not pre-defined, but rather evolve over generations to better suit the given task. Our 3D model supports zoom and rotate so that the user can efficiently observe and explore the entire network or substructures within it. Two modes of operation, interactive and image rendering for video, allow the user to either interactively examine the network throughout the simulation or define preset interactions to be rendered for high frame rate videos. A clock displays runtime (in network time units) throughout the simulation, and the user may interact with the simulation using the play, pause, and clock reset buttons. Time unit duration is adjustable so that the user can shift from an overview of network activity to a fine-grained examination. It is possible to provide user interface elements to allow the user to adjust visualization parameters such as opacity, color scheme, and visual simulation rate, and the addition of these interactions will be addressed in Chapter 7.

Neurons are represented as spheres all of the same size, with color used to differentiate between input, hidden, and output neurons. Alternatively, size, shape, or other distinguishing features of the visual representation of the neuron can be used. Synapses are depicted as lines between neurons with cones at the output end to indicate direction of the synapse. Shading or variation in visual line thickness could also be used to indicate direction. Synapse color encodes positive (excitatory) versus negative (inhibitory) weight, and stroke weight represents the magnitude of synapse weight. Multiple color schemes are available to suit various media.

It is important to note that much of the network's behavior is governed by inhibition of activity (that is, keeping neurons from firing rather than causing neurons to fire). This is true in many different task types, but it is especially true in this task example, in which the network must not fire in approximately 90 percent of the input cases (because the network should identify one digit type out of 10 possible digit types). This type of activity is much harder to track using conventional analysis methods, but it is vital to understanding how each network operates. A major advantage of our existing visualization tool is that it allows us to observe the propagation of charge along the synapses, which are clearly either excitatory or inhibitory, and to see precisely how different input events affect the behavior of the rest of the network.

5.2.2 Highlighting and Filtering

Animated highlighting of activity and modifiable visibility modes leverage pre-attentive processing to facilitate the rapid identification of patterns in network behavior (see Chapter 2 for more on pre-attentive processing and pattern recognition). The tool represents the activity of a network on a specified input by highlighting elements in a contrasting color as events occur on them. Events of interest include neuron fire events and add-charge events on synapses. During the event time window, the size of an element (neuron radius or synapse stroke weight) is increased for further emphasis. Detail views within the visualization also depict charge propagation along each unit of the synapse using highlighted spheres that are smaller than neurons. When visualization of charge propagation is enabled, a longer time unit duration (at least 100ms per time unit) is enforced so that the visualization is comprehensible.

Adjustable visibility modes may also be utilized to filter out details of the simulation that are not relevant to particular tasks. Three visibility modes—invisible, "ghost", and full visibility—allow the user to eliminate visual clutter and draw attention to elements of interest. Invisible elements are not rendered at all, ghost elements are rendered at an opacity of 20%, and fully visible elements are rendered with 100% opacity. Other opacities can be used. In the current version of the tool, the user may set the default visibility of elements (visibility at the beginning of a simulation) to any of the three modes. The simulation may be adjusted to retain the default visualization mode for all elements throughout the simulation, to set elements to be invisible if they are not active within the simulation timeframe, or to transition the elements from default to full visibility as they become active ("visibility upon activity"). Future iterations of the tool will allow users to toggle visibility of any element at any point during the simulation. Chapter 7 discusses the proposed additions to the tool in detail.

5.2.3 Example Network Visualizations

The basic features available at this time in the visualization tool are depicted in Figure 5.2. The runtime (in simulation time units) and buttons for interactivity are at the top left-hand corner of the image in this view, and the view of the network absorbs the center of the screen. Other locations for user controls may be utilized. The column of green spheres at the left side of the network are the input neurons, and the single orange sphere at the far right of the network is the output neuron. The hidden neurons are the the teal spheres between the input and output neurons. Positive weighted synapses are lines colored blue, while negative weighted synapses are in red-orange.

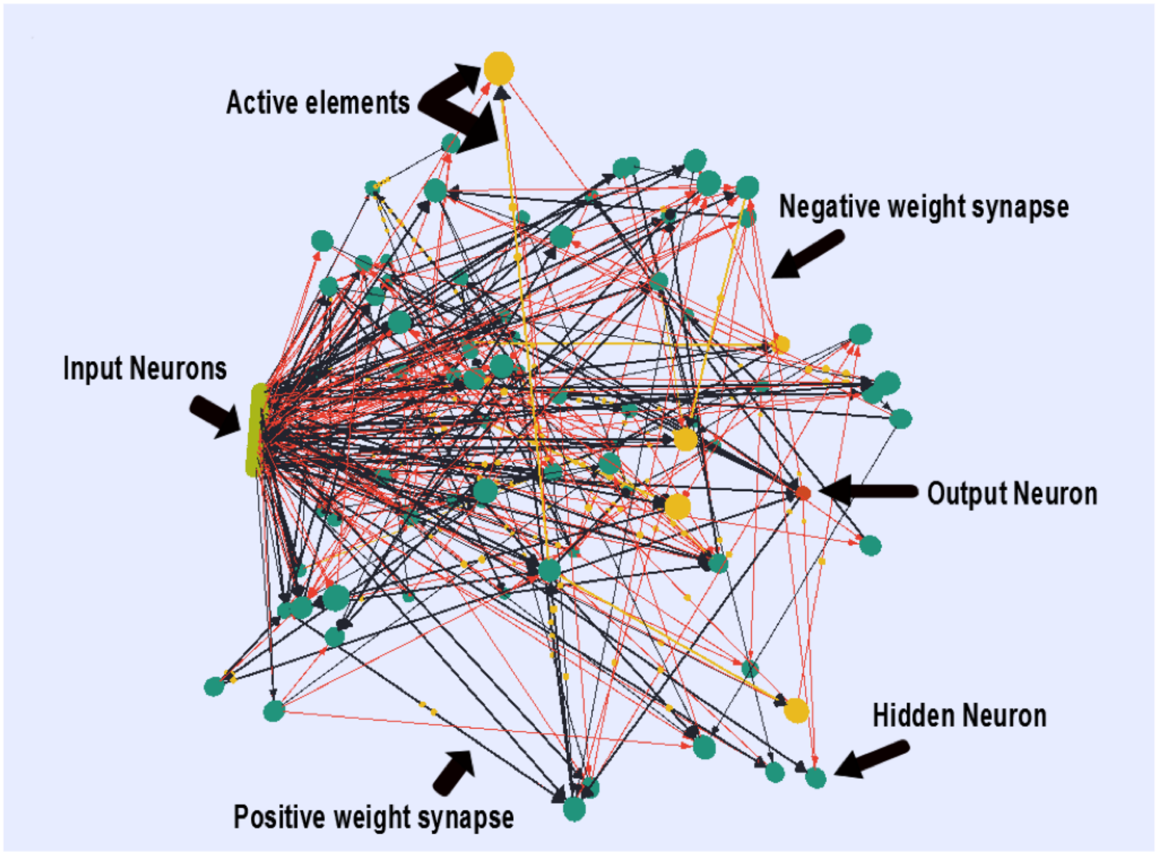


Figure 5.2: An example NIDA network shown using the visual analytics tool. This network is trained to recognize handwritten digits of the number 7, shown at time unit 237 in a simulation processing an image that shows the digit 7.

Other visibility settings are shown in Figure 5.3. The simulation shown in this image had a default visibility of ghost for hidden neurons and synapses, so the full opacity lines and teal spheres represent network elements that have already been active during the simulation—synapses that have transferred charge to neurons and neurons that have fired. The lines and spheres with 20% opacity depict elements that have not yet been active during the simulation. Elements highlighted in bright yellow are those that are active at the current time step—synapses at the point when the charge they have transferred reaches the receiving neuron and neurons currently firing. The smaller yellow highlighted spheres along various synapses stand for the points of charge propagation along the given synapses. Each charge propagation point results from a distinct event, the firing of the neuron from which the synapse originates. Should the charge propagate to the receiving neuron within the timeframe of the simulation, it will effect a unique event of addition or reduction of charge in the receiving neuron. Unless otherwise noted, all printed network figures in this work use the same color scheme.

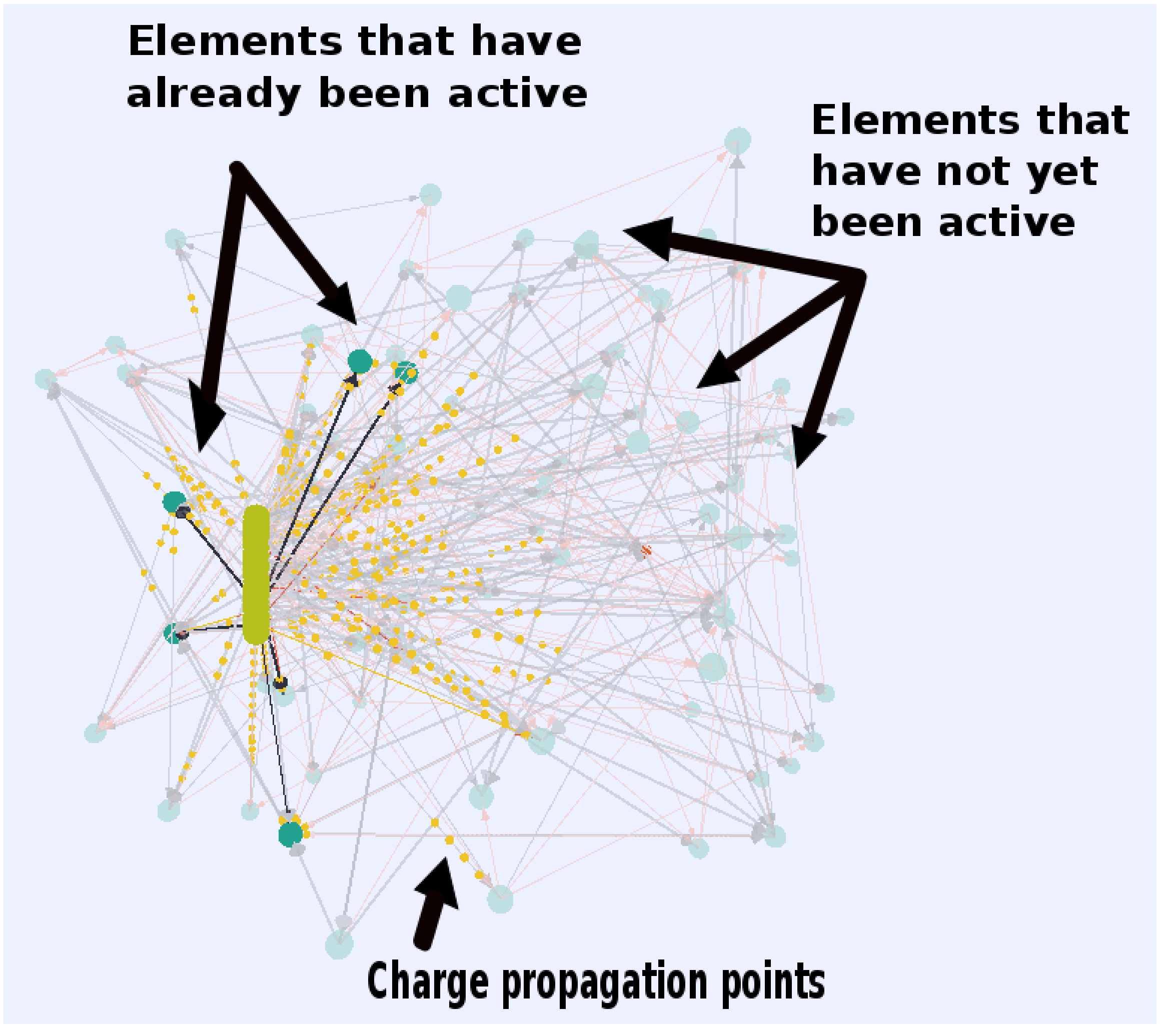
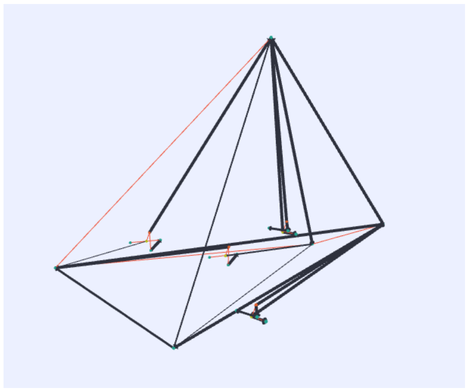

Elements that have already been active
Elements that have not yet been active
Charge propagation points

Figure 5.3: Example network set to use visibility upon activity with a default visibility of ghost for all elements.

5.2.4 Scaling and Visualization of Subnetworks

As an enhancement to other interactive features, we have implemented adjustable scaling within the network so that the relationships in more compact networks and substructures can be examined. The exploration of dense networks requires scaling in addition to zooming because some networks allow for neurons to overlap within a single unit of space. In order to view relationships between neurons so closely situated, the space that each neuron occupies must be reduced in proportion to the space of the network overall.

In some networks, substructures are actually complete networks embedded within larger networks. An affective system is an example of such a subnetwork. Affective systems have their own fitness functions distinct from those of the networks in which they are embedded, yet they are designed to cooperated with the larger network. For example, the larger network could represent an agent whose goal is to survive as long as possible in the environment. Affective systems can be embedded within the network to symbolize fear—to help it avoid predators—and for curiosity—to more effectively explore the environment in search of food. Embedded subnetworks may have different granularities than their larger networks. In one realization, the larger network (the agent) has a granularity of 1.0, meaning that neurons can exist no closer than 1.0 unit of space from each other. The agent network exists in a 100 x 100 x 100 grid. In this realization, examples of which are shown in Figures 5.4 and 5.5, the affective system subnetwork has a granularity of 0.1 and exists in a 1 x 1 x 1 grid.


(a) View of full network.

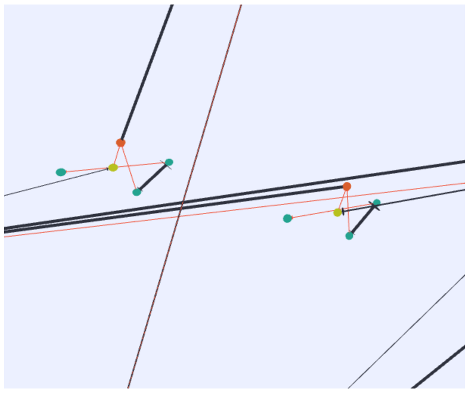

(b) Zoomed view of embedded subnetworks.

Figure 5.4: Example network with embedded affective systems as subnetworks.

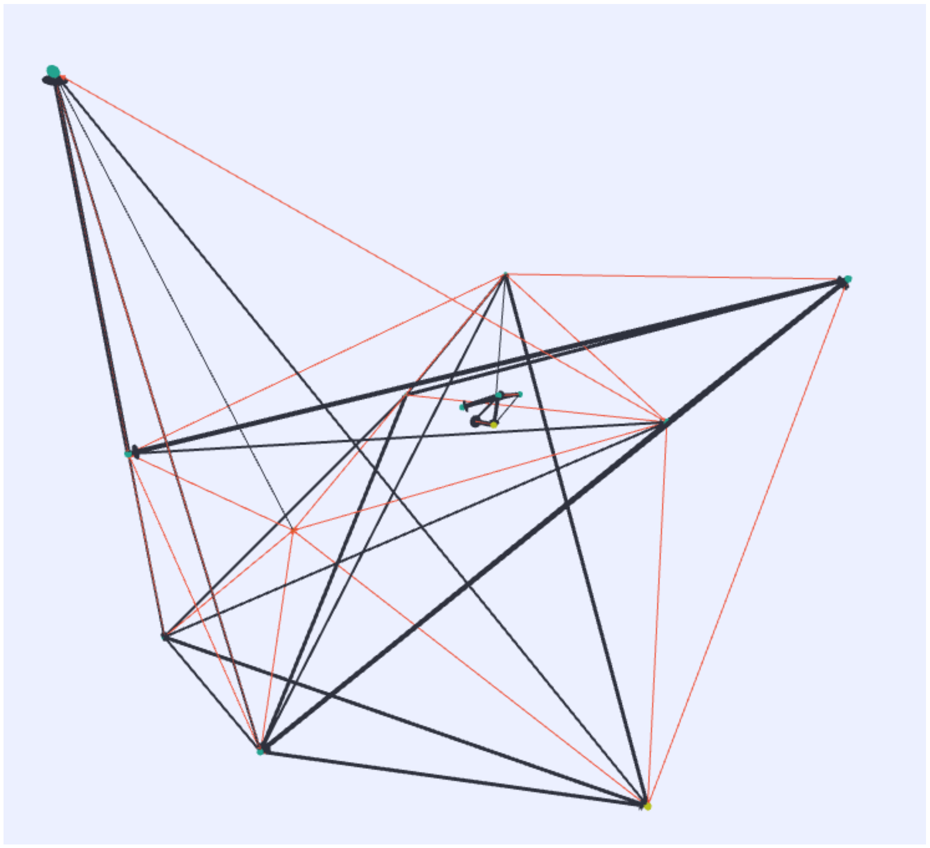


Figure 5.5: Large network with granularity of 1.0 with embedded subnetwork of granularity 0.1.

There are multiple ways to visualize these networks and subnetworks of dramatically different scale. Figure 5.4a shows the example agent network with two embedded affective systems for fear and two for curiosity. A zoomed-in view of the curiosity subsystems is shown in Figure 5.4b. Figure 5.4 demonstrates the capability of the visualization tool to scale networks to the smallest level of granularity, allowing users to view the entire embedded networks within the context of the larger network. Another example of this embedded view for subnetworks is shown in Figure 5.5.

An alternative technique for visualizing embedded networks is the use of placeholders. Figure 5.6 illustrates this method, depicting the same agent network from Figure 5.4 with placeholders instead of embedded subnetworks. Placeholders are depicted as cubes to differentiate them from neurons, and color is used to distinguish between types of embedded network. This technique still allows for the combination of details and context in the visualization, since the affective system networks may be shown in additional views. Determining the more appropriate of these two methods may depend on the user's task or visual query. Both features are available.

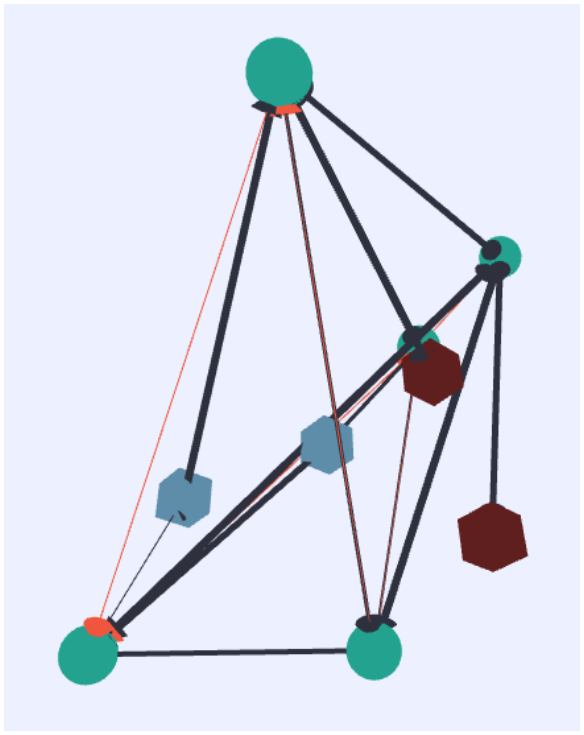


Figure 5.6: Agent network from Figure 5.4 with placeholders representing each affective system.

The placeholder method may also be used to visualize collaborative tasks among teams of networks. As presented in Figure 5.7, this technique also uses color to distinguish between types of networks. As with embedded networks, the networks represented by placeholders may be shown fully in additional views.

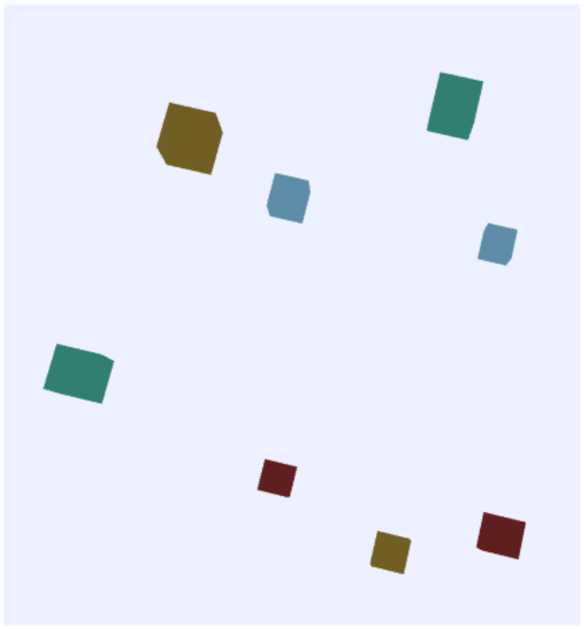
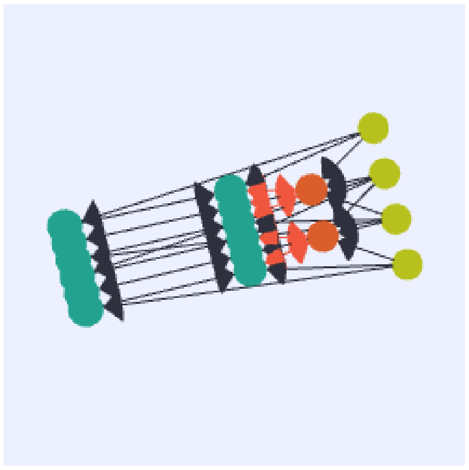


Figure 5.7: A team of networks shown using placeholders for each network.

Scaling has enabled the visualization of compact hand-tooled networks in addition to helping facilitate exploration of embedded networks. Although NIDAs are primarily developed using evolutionary optimization, they can be manually created. NIDA developers are producing a library of hand-tooled networks and substructures with known functionality that can be inserted into larger networks. Hand-tooled networks tend to be more compact than their evolved counterparts, even given the same granularity. Figure 5.8a shows one such hand-tooled network, which is capable of recognizing vertical lines. A larger network, shown in Figure 5.8b combines multiple vertical line recognizers in order to identify vertical lines in a larger field. The adjustable scaling features in the visualization tool allow users to explore networks of various granularities and various neuron placement densities.


(a) Hand-tooled vertical line recognizer network.

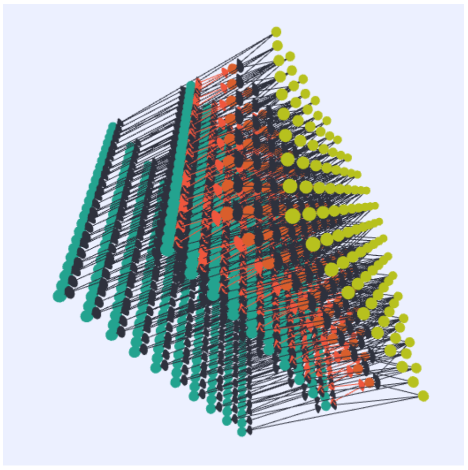

(b) Network composed of arrays of vertical line recognizers.

Figure 5.8: Hand-tooled networks shown using scaling features of the visualization tool.

5.3 Simulation Videos

In order to better represent the functionality of the visualization tool, videos of example NIDA network simulations are linked to this work and can be found in the University of Tennessee's online catalog. Attachment 1 shows the processing of three input images of the digits 0, 5, and 2 by a network trained to recognize the digit 0. This video was produced using the video rendering operating mode to allow comparisons of the network's behavior on different input images. Attachment 2 shows a different handwritten digit classifier network using the interactive exploration operating mode.

The color scheme used for videos is distinct from the one used for print images, since a darker background allows for better perception of events during the simulation. In the videos, input neurons are colored yellow, hidden neurons are teal, and the output neuron is red. Positive weighted synapses are blue and negative weighted synapses are orange. The highlight color for active elements and charge points is white. The default visibility mode for the videos is ghost, and visibility upon activity is used, showing the elements as fully visible after they have been active during the simulation time frame.

5.4 "Similarity" and Useful Substructures

5.4.1 Background: Evolutionary Optimization Method

One of the most pressing challenges facing the developers of NIDA networks is the design of the evolutionary optimization process so that it is highly efficient. The evolutionary optimization, described in detail by Schuman et al. [28] is robust and produces high-performing networks, but it is the bottleneck in the creation and training of networks due to the many generations required for successful evolution.

In a current realization of the method, the evolutionary process is initialized randomly. Two operations, crossover and mutation, are used to create child networks from parent networks. High-performing networks, as determined by a fitness function for a particular task, go on to form the next generation of parent networks. Mutation operations occur at randomly determined points, and they include the removal or addition of neurons or synapses to a network or changes to parameter values. The crossover operation splits two randomly chosen parent networks and exchanges parents' subnetworks to create two new networks. Figure 5.9 shows the evolutionary optimization process.

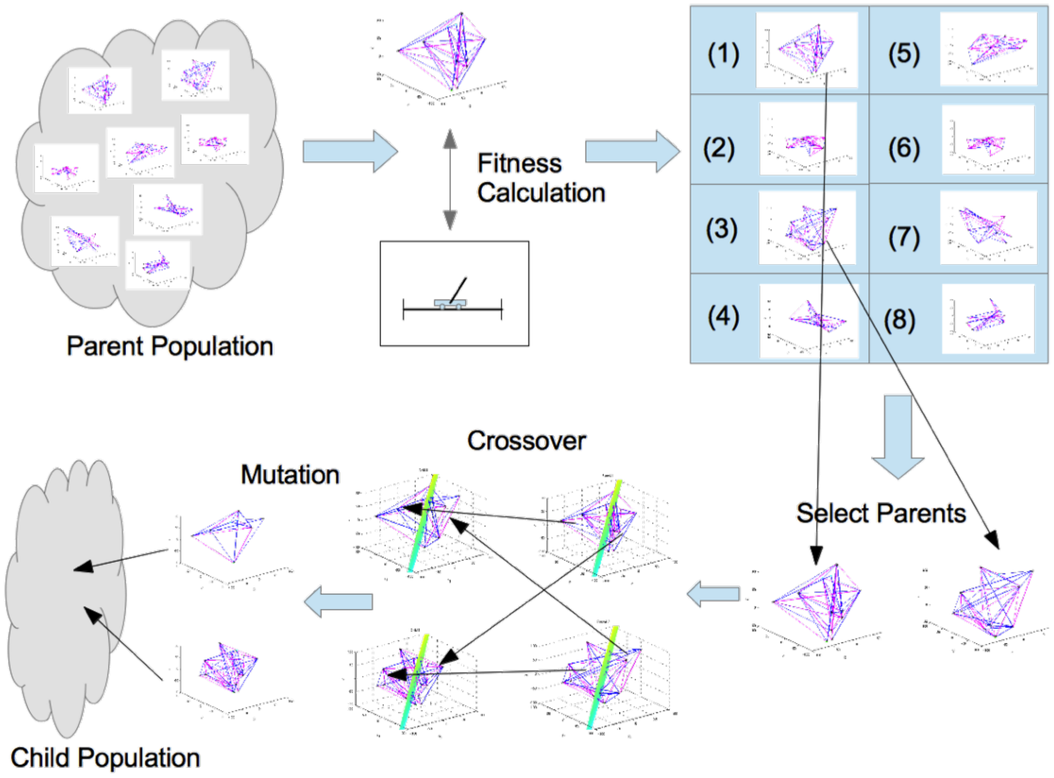


Figure 5.9: A summary of the evolutionary optimization method, showing the two operations used to create child networks from a parent population and the fitness function as the means by which a network's performance is evaluated. (Figure created by Catherine Schuman and used with her permission.)

5.4.2 Similarity

We hypothesize that the evolutionary optimization method could be improved and accelerated by leveraging knowledge gained from networks that behave "similarly." Similarity may be defined in different ways. For a given network type with specified input and output connections, we define three different types of similarity:

1. Input/output (behavioral) similarity: Given similar input event sequences, the two networks produce similar output event sequences. Input/output similarity does not measure the similarities in the graph structures or parameter values of the two networks.

2. Structural similarity: The two networks have similar graph structure. Optionally, similarities in the parameter values may be measured. Structural similarity does not measure the similarities in the input/output behaviors of the two networks.

3. Information flow similarity: Information flow similarity assumes substantial structural similarity so that paths in the two networks can be associated and compared. Given two networks N_1 and N_2 defined by their graphs, identified inputs and outputs, and parameters, and given a set P of pairs of associated paths (p_1, p_2), where p_1 and p_2 are paths in the graphs of networks N_1 and N_2, respectively, information flow similarity is a function of the time sequences of events occurring on the synapses and neurons of the identified associated paths.

While these three definitions of similarity are related, they are not nested. Information flow similarity assumes some structural similarity, but a degree of input/output similarity is assumed only if the set of associated paths includes paths that contain the inputs and outputs of the networks or the inputs and outputs of specified substructures. Input/output similarity and structural similarity are different. The networks can be viewed as directed graphs with information associated with the neurons and synapses (parameters and events that occur as a function of time in response to inputs). Structural similarity ignores the event sequences, and input/output similarity ignores the structure of the graphs (other than its inputs and outputs). Information flow similarity presumes prior computation to determine associated paths in the networks' graphs and uses that information to analyze the similarities of event sequences associated with these path pairs.

A user may choose how similarity is defined and computed. The functions that quantify similar input/output behaviors, similar graph structures, and similar information flows may also be determined by the user. In addition, information flow similarity may not quantify similarity on all paths of two networks; the reader is left to decide which pairs of associated paths are to be analyzed. In addition, all three types of similarity may be applied to a subnetwork; it is not essential that entire networks be analyzed. This is desirable because we may wish to identify substructures of networks that have particular utility for a given application.

Behavioral similarities that depend upon events as a function of time also depend upon the inputs used to excite each network's behavior. Input event streams may be grouped in classes, and it is desirable to determine a network's response to multiple event streams of a class in order to statistically characterize its behavior when presented inputs from that class. While a network's response to a specific input event stream may be interesting, it is more important to characterize the network's response to all inputs of a class, and to do this over all classes of interest. We note also that a lack of response is often equally important. Behaviors internal to a network in response to input event streams may suppress outputs—for example, when a network is designed to recognize only inputs of a specific class by generating outputs within a specified time interval.

We believe that one approach to improve the performance of the evolutionary optimization method is to identify and exploit behavioral similarities across sets of networks. Ideally, we wish to identify networks or subnetworks that respond in a desired way to stimuli and can be reused in the design of new networks. We believe it may be possible to adapt an evolutionary optimization process to improve the rate at which it produces satisfactory networks by incorporating randomly selected network change operators that utilize such substructures.

Information flow similarity is a more complex attribute of networks than input/output similarity and structural similarity. Similarity of information flow assumes at least substantial structural similarity, but it also requires behavioral similarity at a finer grain. One way to define information flow similarity is recursive behavioral similarity. That is, two networks that demonstrate similarity of information flow are behaviorally similar at the highest level in that they produce similar outputs for similar inputs. Additionally, they may exhibit behavioral similarity for certain substructures within each network. In order to exhibit information flow similarity, these networks need not have behavioral similarity at every level, but they should have behavioral similarity at multiple levels, including in one or more substructures. The degree of information flow similarity may be defined by the number of levels of behavioral similarity that exist between networks.

Similarities may also be defined for the same network in response to different input event streams, either within a single input class or multiple classes. Similarities may be identified in various sub-problems:

1. Similarities that appear in a single network over many inputs of the same class,
2. Similarities that appear in a single network over many inputs of different classes, and
3. Similarities that appear in multiple networks in response to:
    (a) a single input,
    (b) many inputs of the same class, and
    (c) many inputs of different classes. (This type of similarity may be used to infer classes of networks that have similar behaviors.)

It is expected that similar behavior will correlate with a similar flow of information internally for some, but not all, networks. Stated differently, networks may successfully employ multiple approaches to the solution of the same problem. We also recognize that similar behavior and information flows should be observable not only between complete networks, but also for substructures within networks. In order to take advantage of these “useful” substructures, we propose the addition of another operation to create child networks from parents. We call this operation “surgery,” since it can be used to remove or implant specific useful substructures from parent networks in a future generation of child networks. The problem that the remainder of this chapter seeks to explore is the identification of such substructures.

5.4.3 Useful Substructures

We hypothesize that useful substructures are substructures that exhibit behavioral similarity over multiple inputs in at least one network. For example, on a handwritten digit classification problem, a substructure could be useful in recognizing a particular attribute of the digit that a network has been trained to recognize, such as the continuous curves of a zero. Substructures for both successful and unsuccessful networks should be explored. It is anticipated that the identification of structural and information flow similarities will facilitate and/or complement the discovery of behavioral similarities between useful substructures.

For the surgery operation to be useful in the long-term, it must be automated within the evolutionary optimization method. We anticipate that visualization tools will help drive the development of automated methods by revealing patterns that may otherwise have been difficult to observe. The NIDA visualization tool has already facilitated the identification of potentially useful substructures in networks trained for handwritten digit classification. The current methods for substructure discovery are activity-based and event-based, and are described in more detail by Drouhard et al. [7] and in the remainder of this chapter. In one instance, the tool facilitated the discovery of a highly active substructure, which further analysis showed to be active over all zero-digit inputs in a network trained to recognize the digit zero. It also allows for the tracing of "causality paths," or paths of neurons and synapses whose activity contribute to specific events within the behavior of a network over a single or multiple inputs, as described in Section 5.5. An example showing the procedure for the surgery operation on a causality path is shown in Figure 5.10. We discuss the anticipated improvement and expansion of these capabilities in Chapter 7.

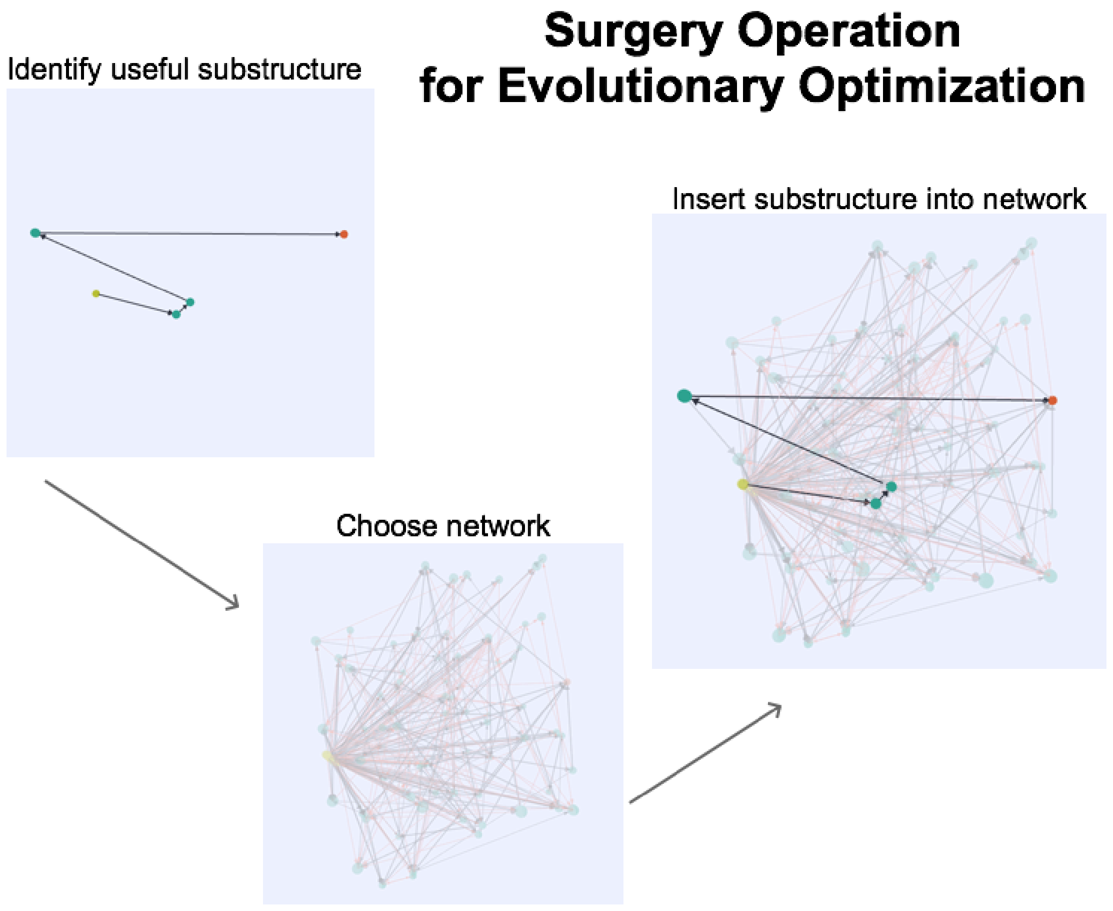

Surgery Operation for Evolutionary Optimization
Identify useful substructure
Choose network
Insert substructure into network

Figure 5.10: Example of the surgery operation, showing the implantation of a useful substructure into a network trained for the same task as the network from which the substructure was drawn.

5.4.4 Activity-based Identification of Substructures

Visual simulations on a handwritten digit network trained to recognize the digit 0 yielded the discovery of one interesting substructure. For a network N, trained to recognized the digit 0, the three-neuron substructure s shown as the three active (yellow) neurons in the highlighted region in Figures 5.11 and 5.12 was observed to be highly active throughout the processing of multiple input images of the digit 0. To better understand the activity of s in relation to other neurons within the network, we performed an analysis of the activity of all neurons in the network over all input images of the digit 0, the results of which are shown in Figure 5.13. The three neurons contained in s were more active than all other neurons in the network by a wide margin. Based on this data, we speculated that s is a significant substructure to the behavior of N during the processing of 0 input images, and therefore, that it might be a useful substructure to implant in other networks trained to recognize the digit 0. However, when we analyzed the activity of neurons in N during the processing of input images of non-0 digits, the results confounded our hypothesis. As shown in Figure 5.14, the neurons of s are highly active during the processing of non-0 digits as well. It appears that this substructure is significant to the behavior of N in general, not only during the recognition of 0 input images. Thus, it may not be useful substructure to replicate in other networks.

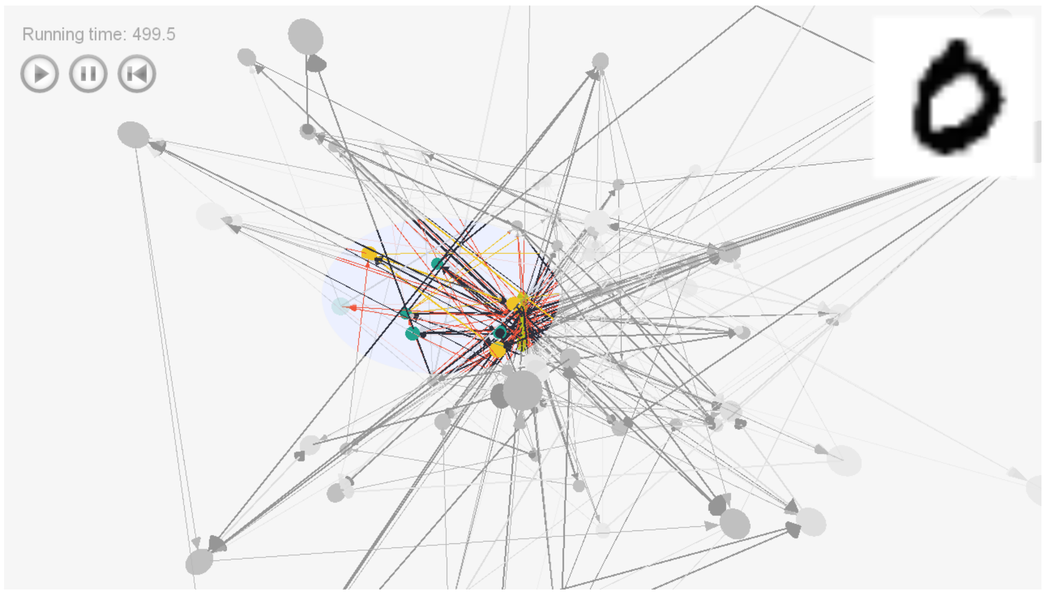


Figure 5.11: A three-neuron substructure highlighted within a handwritten digit network trained to recognize the digit 0, shown during a simulation of the network processing an input image of a 0. The substructure contains the three active (yellow) neurons in the highlighted region.

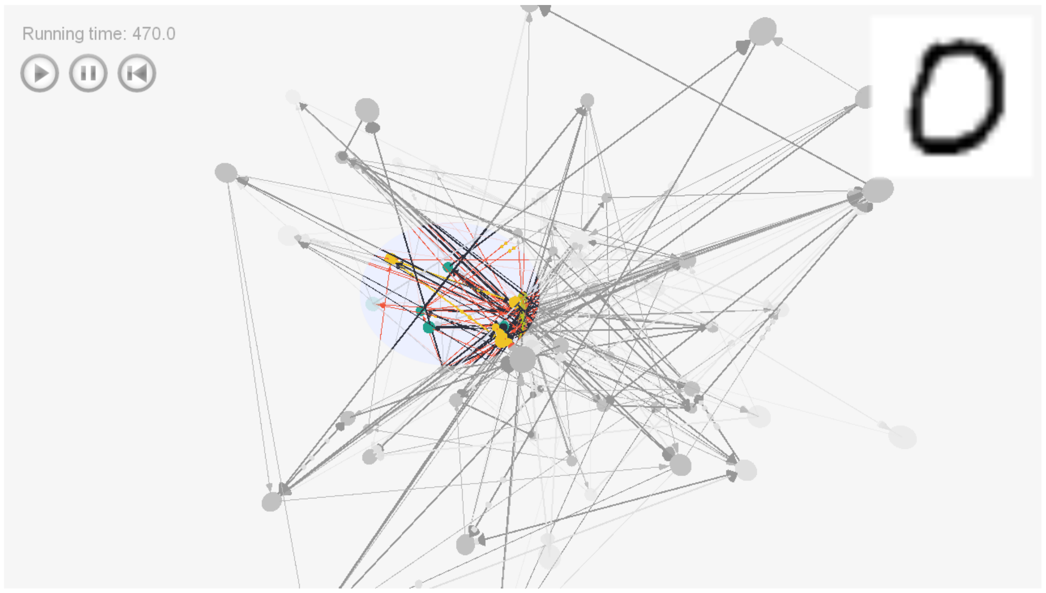
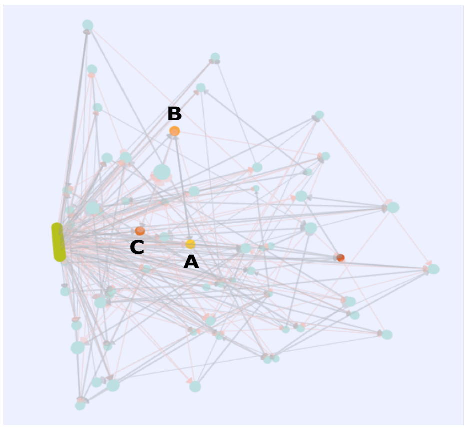
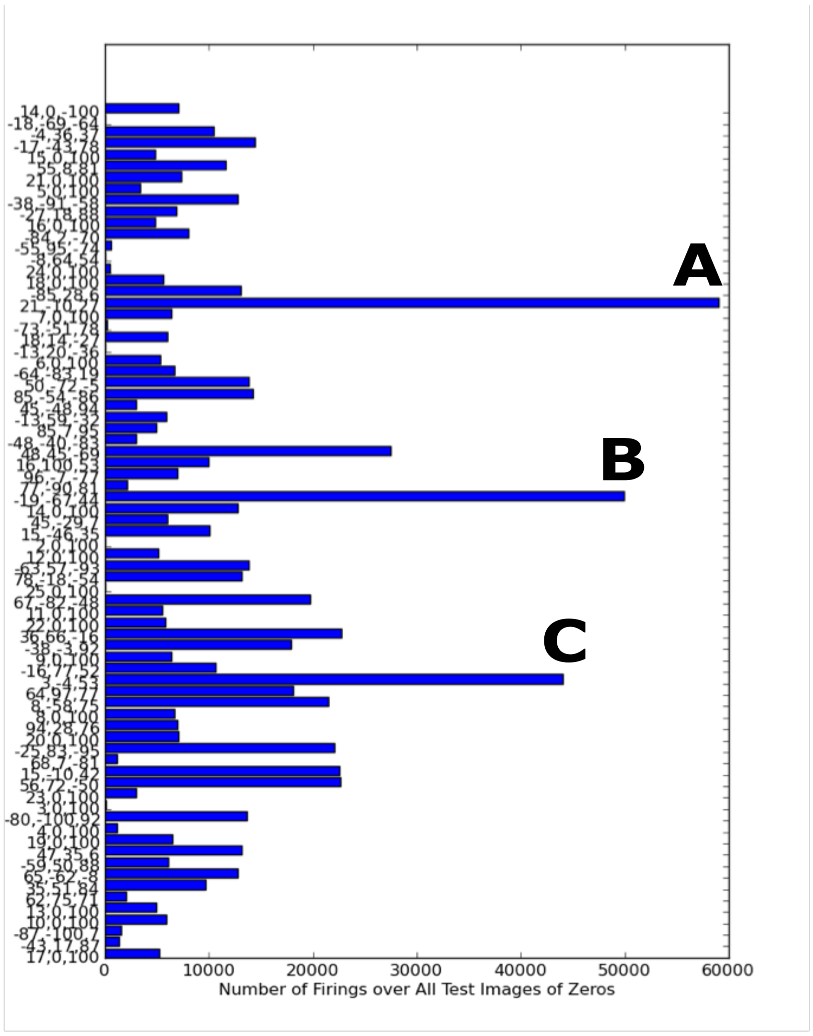


Figure 5.12: The same three-neuron substructure within the same network shown in Figure 5.11. This image is drawn from a simulation of the network processing a different input image of a 0.

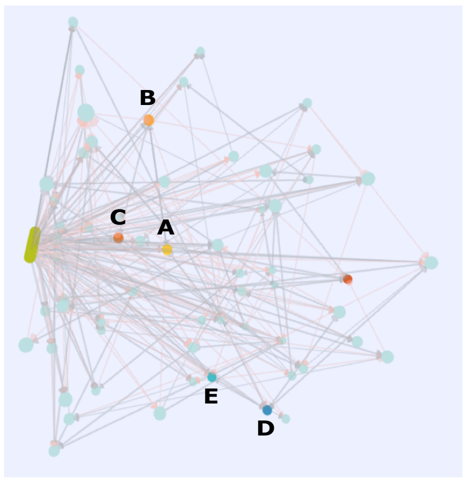


(a) Labeled neurons of substructure s.

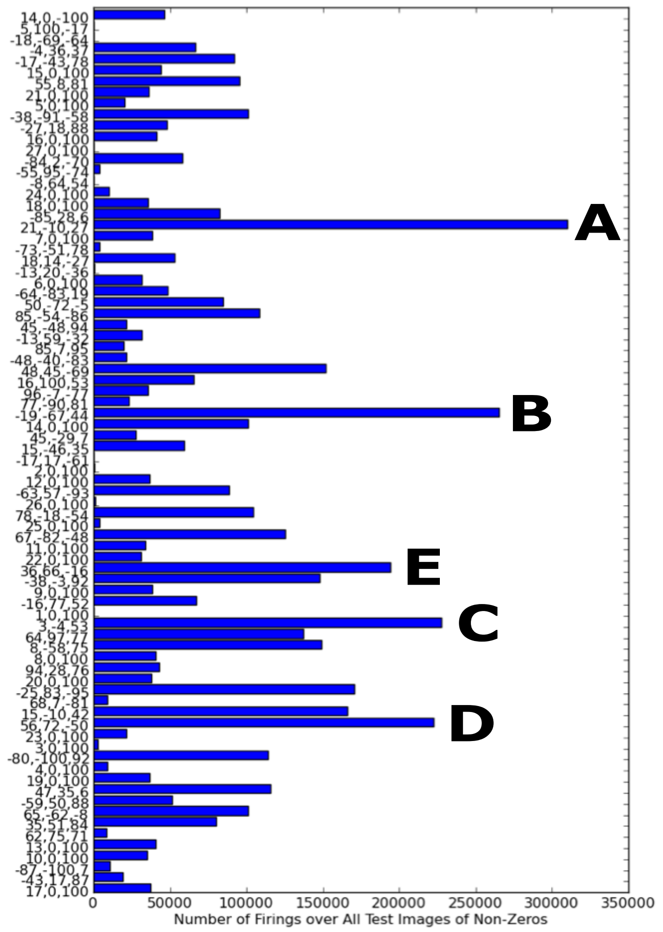


(b) Neuron fire event statistics for input images of digit 0.

Figure 5.13: Neurons in substructure s and firing statistics for digit 0 input images.



(a) Five most active neurons for non-0 input images, including substructure s.



(b) Neuron fire event statistics for input images of non-0 digits.

Figure 5.14: Active neurons, including substructure s, and firing statistics for non-0 input images.

Substructures such as this one can be identified easily by determining the most active neurons in the network, but this method may be too simplistic to improve the evolutionary optimization (EO). When the EO was extended to automatically identify the top 5% most active neurons and implant them into child networks as an additional operation, EO did not improve overall. In some cases, the addition of this operation significantly improved EO, in other cases it significantly harmed performance, and in some cases it had little or no effect.

5.5 Causality Paths

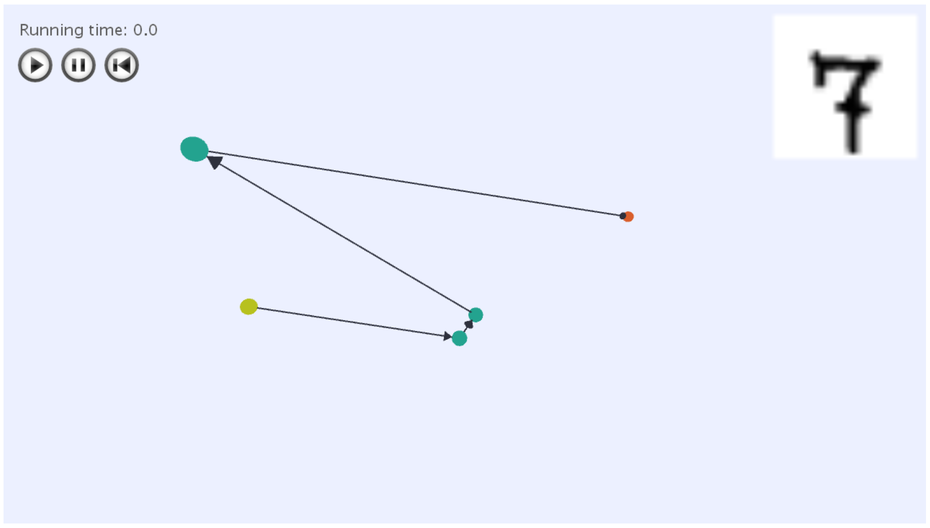
5.5.1 Event Summaries and Path Tracing

In an effort to identify substructures that will be useful in other networks, we extended the visualization tool to target substructures whose behavior we may wish to replicate. Besides animating all of the activity of the network on a given input, the visualization tool has an alternate mode that allows for the isolated viewing of specific events and the activity that leads to them. This mode requires the compilation of detailed event summaries for each element (neuron and synapse) on the network. An event summary consists of every event e that occurs on a given element, along with other events throughout the simulation that contribute to the occurrence of e and those events triggered in part by e. For example, a neuron fire event f is defined as being affected (and eventually triggered) by every synapse charge event that reaches it after the previous firing and before the occurrence of f. A synapse charge event is defined as a single event that incorporates delay by virtue of the synapse's length. Synapse charge events are caused by the firing of the neuron from which the synapse emanates and affect the subsequent firing (or absence of firing) of the neuron to which the synapse connects. Using the event summaries created, we are able to create timelines of causes and effects for each event e. We can also visually trace from e back to the initial input neuron pulse(s) that trigger(s) them or forward through all of the events to which e

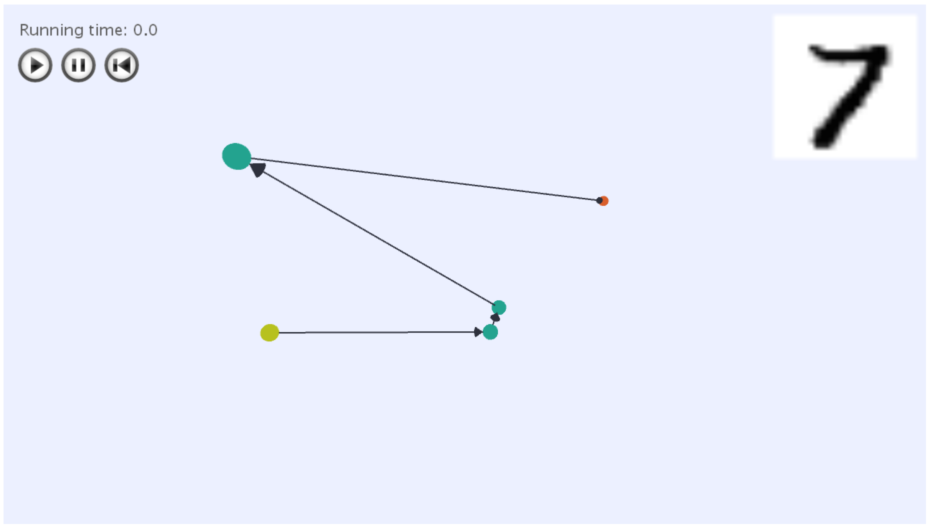
contributes during the simulation timeframe. This functionality enables the user to identify portions of the network's structure that participate in computations leading to a particular outcome, as well as portions of the network's structure that are affected by a particular event.

Using this mode of the visualization tool, we have traced important events in the network back to the initiating events on input neurons. We refer to these paths as "causality paths." The activity along the path can be animated in the same way as standard network activity in order to trace the precipitating actions from input neuron pulse(s) to the occurrence of the event itself. One experiment with these paths explores the differences in the output neuron activity between input images of the digit d that a network has been trained to recognize and input images of digits other than d. Of particular interest are images of non-d digits that share certain characteristics with d.

Figure 5.15 is a path extracted from the activity of network S, a handwritten digit recognizer trained to recognize the digit 7, shown during the processing of two different input images of 7's. The figure shows the first firing of the output neuron during the final time window, signaling recognition of the digit 7. Though these fire events on the output neuron occur at different times for the different inputs, the path is the same. In contrast, Figure 5.16 shows a path drawn from the same network during the processing of an input image of the digit 2, which has some similar features to images of sevens. The path in Figure 5.16 traces the final firing of the output neuron in network S, which occurs prior to the final time window, indicating correct recognition of a non-7 digit.



(a) Path extracted from the activity of a network trained to recognize the handwritten digit 7. The path has been traced back to the input pulses from the firing of the output neuron during the final time window in the processing of an input image of the digit 7.



(b) Same path through the same network shown in Figure 5.15a, processing a different input image of a 7. The event traced by the path is the first firing of the output neuron within the final time window, which signals that the network correctly identifies the image as a 7.

Figure 5.15: Causality paths through a network trained to recognize the digit 7, tracing back from fire events on the output neuron.

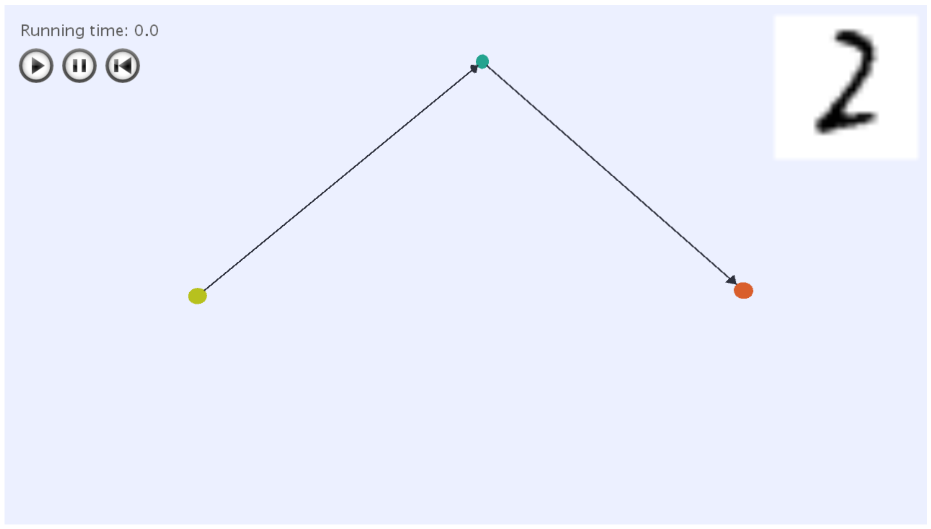


Figure 5.16: Causality path from a network S (discussed in Section 5.5) trained to recognize the digit 7 processing an input image of the digit 2. The event traced by the path is the last output neuron fire. The fire occurs prior to the final time window, indicating that the network recognizes that the image does not show the digit 7.

The network behavior was similar for multiple input images of the digit 2, as well as for other non-7 digits. For several input images of 2's, the final firing of the output neuron could be traced back to the input pulse along the same relatively short two-segment path. The final firing event primarily propagated charge along inhibitory synapses. Other input images, such as those of the digit 1, triggered different activity, but the paths to the final firing tended to be short and to trigger more inhibitory behavior. An example causality path for the digit 1 is shown in Figure 5.17. This figure traces the path of the output neuron back along a single segment path to its initiating input pulse. It also traces forward the activity triggered by the fire event on the output neuron. This activity is primarily inhibitory. Causality paths for many other input images of the digit 1 were similar. The paths for correct recognition of the digit 7 tended to vary more, but were longer overall, as could be expected since the fire to indicate recognition of d must occur within the final 50 time units. The variation in paths for images of 7 may be attributable to the variations in ways 7's can be written, as shown in Figure 5.18.

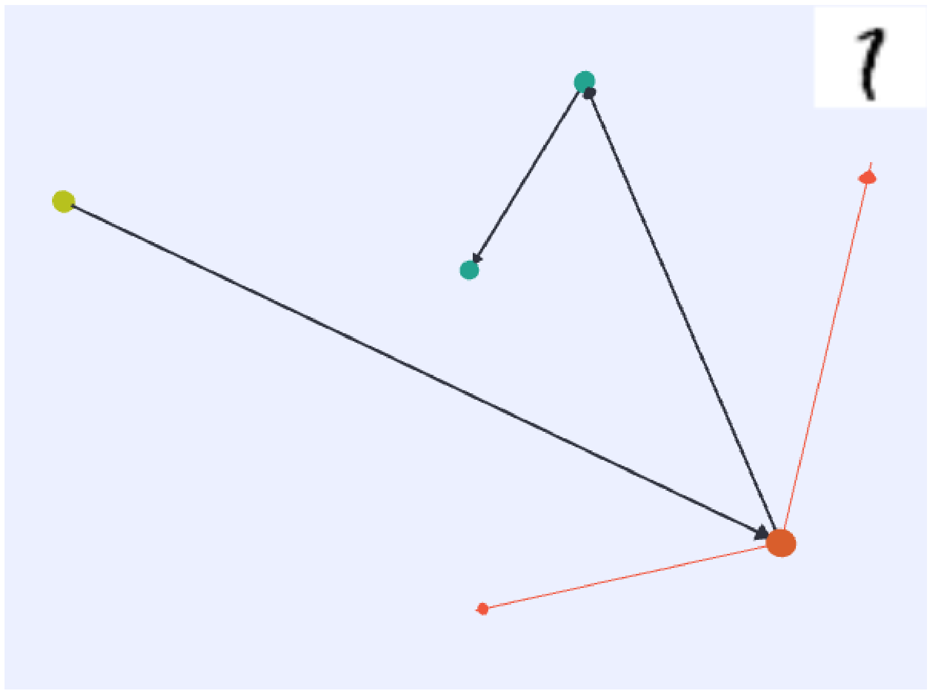


Figure 5.17: Causality path from network S trained to recognize the digit 7 processing an input image of the digit 1. The event traced is the final firing event of the output neuron, which indicates the correct recognition of a non-7 digit, since it occurs prior to the final time window. The fire event is traced backward to the initiating input pulse, as well as forward to the triggering of primarily inhibitory activity.

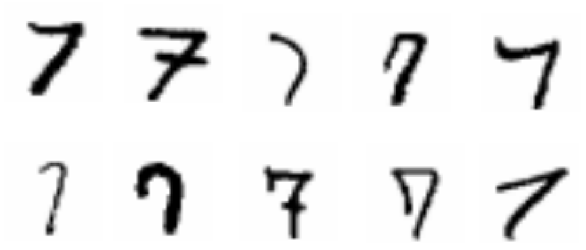


Figure 5.18: Example images of written 7's. The variations in the way 7's are drawn may contribute to the variation in causality paths that occur.

The causality paths for network S provide further intuition about how networks of this type operate. Based on these results, we can speculate that shorter paths to the final firing of the output neuron for 1's and 2's, as compared to the paths traced when identifying an image as a 7, indicate the relative ease of identifying an image as a non-d digit, rather than a d digit. That is, it is easier (and requires less complicated structure) for the network to determine that an image is not of a digit d than it is for the network to definitively say that the image is of a d. Table 5.1 gives the classification results of one of these networks in isolation (a network trained to recognize images of 7's). In particular, this table shows that for images of non-7 digits, excluding 9's, the network achieves higher than 90 percent accuracy (that is, does not fire in the last time window for these images), whereas it only achieves around 80 percent for images of sevens. The low accuracy rate for 9's may be attributed to the similarities in the ways 7's and 9's are written.

Table 5.1: Accuracy Breakdown for a Network Trained to Recognize Images of the Digit 7

Digit	**Accuracy**
0	99.4898
1	99.9119
2	97.6744
3	90.8911
4	97.4542
5	92.9372
6	99.791
7	79.3574
8	94.5585
9	77.106

It is worth noting that each of the causality paths presented for network S traced back to a single event on a single input neuron. Since this network scans an input image column-by-column, the traces would seem to indicate that this network decides whether or not the image is of the digit 7 based on a single pixel, yet we know that the decision must be more complex. Rather, we hypothesize that the longer, more complex paths toward recognition of d digits are always initiated by one or more input neuron. If the network determines—using the shorter, simpler paths—that the input image being processed is of a non-d digit, inhibitory activity is triggered, preventing the d digit recognition path(s) from completing. The extension of the visualization tool to encompass causality "subnetworks" has so far supported this hypothesis.

5.5.2 Causality Subnetworks

Causality subnetworks represent the causality paths of multiple events of interest, and can include paths from a network drawn from the processing of different inputs. Since these networks represent multiple sequences of events, the visual encoding is slightly different than the standard visualization. Neurons are still depicted as spheres, and synapses as lines whose directionality is indicated with cones. However, in these visualizations, neuron size and synapse stroke weight encode frequency of the element's appearance in a causality path. Sample causality subnetworks for networks trained to recognize the digits 2, 4, and 6, respectively, are shown in Figures 5.19, 5.20, and 5.21. The subnetworks trace causality paths for all output neuron firing events in the final time window for all input images of the digits that each respective network was trained to recognize. In other words, they trace the causality paths for all of the network's behavior that indicates successful recognition of the d digit. Larger size and stroke weight indicate the that neurons and synapses appear in these paths more frequently.

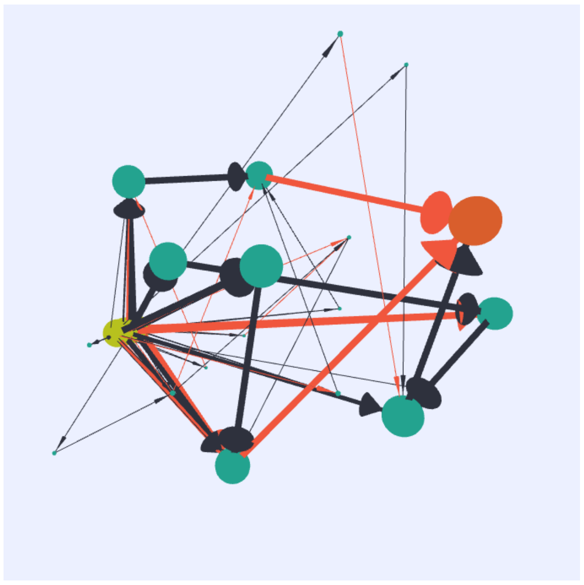


Figure 5.19: Causality subnetwork from a network trained to recognize the digit 2, tracing all output neuron firings in the final time window for input images of the digit 2.

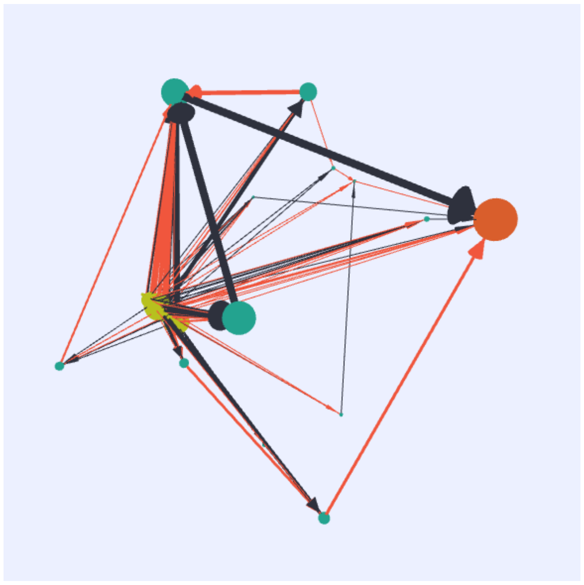


Figure 5.20: Causality subnetwork from a network trained to recognize the digit 4, tracing all output neuron firings in the final time window for input images of the digit 4.

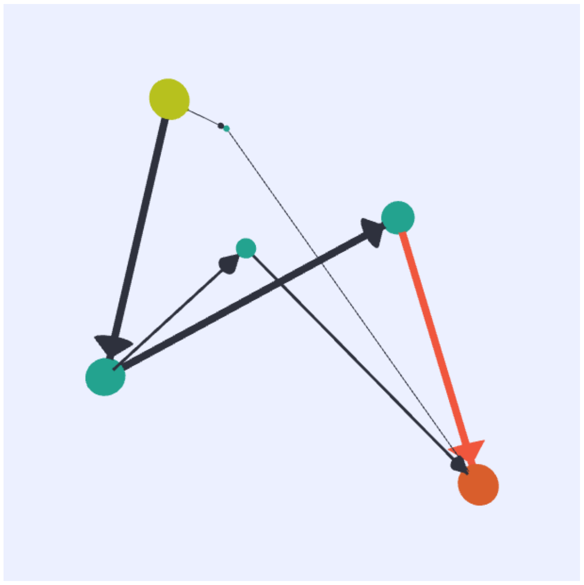


Figure 5.21: Causality subnetwork from a network trained to recognize the digit 6, tracing all output neuron firings in the final time window for input images of the digit 6.

The insight we have gained from causality subnetworks aligns with our previous observations from causality paths. Path tracings of events that indicate recognition of the d digit tend to be longer and have more hops, and the same paths recur frequently for multiple input images of the same class. The difference in complexity of the networks is also intuitive. The digits 2 and 4 may be written in a relatively wide variety of ways (loops vs. no loops, closed vs. open tops, etc.), while the digit 6 is generally written in the same basic form. Understandably, then, the networks trained to recognize 2's and 6's would require a greater number of paths to recognize input images of the d digit. The variation in correct forms of 2's and 4's likely accounts for the greater structural complexity of the causality subnetwork for these digits.

Causality paths and subnetworks are helpful in understanding the structures in a network that are important in producing the functionality of the network. They are another automated way to track useful substructures that may be exploited during the EO method, but unlike the activity-based technique for identification of substructures, causality paths target specific behavior of interest. We plan to extend the exploration of causality paths to allow the user greater interactivity in selecting an event or multiple events to trace and in highlighting different aspects of the activity shown in the trace. The proposed extensions are outlined in Chaper 7.

5.6 Summary

In this chapter, we have described the features available in the NIDA visualization tool and some of the insights that the tool has facilitated. The tool represents three-dimensional NIDA networks in an interactive environment that encourages exploration. It allows user to trace activity of networks, and explore the structure and behavior of networks at various levels of detail.

The NIDA method for evolutionary optimization performs well, but it is not as efficient as is desired. We hypothesize that the method can be expedited through the recognition of networks and substructures whose behavior we would like to emulate in or eliminate from other networks. In other words, we believe that the identification of similar networks or substructures that are either successful or unsuccessful for particular tasks may allow us to improve the evolutionary optimization method. We have proposed to expand the evolutionary optimization method to allow for the implantation or removal of such substructures through a surgery operation, and this technique is currently being incorporated into the NIDA design method. The current iteration of the visualization tool has facilitated some methods for exploration of useful substructures, and we hope to expand these methods in future iterations. The next chapter explores a related visual interface for DANNA networks.

Chapter 6

Two Dimensional DANNA Visual Interface

6.1 Background on DANNA Networks Visualized

Dynamic Adaptive Neural Network Arrays (DANNAs) are a hardware implementation of the NIDA architecture and design method. DANNAs are composed of rapidly reconfigurable neuromorphic elements that may represent neurons, synapses, or other necessary elements. The first implementation, with each element connected to its eight immediate neighbors, is introduced by Dean et al. [6]. Extensions to the DANNA design, including increased connectivity of elements and scalability to larger arrays, are discussed by Daffron et al. [4]. The neuromorphic elements that compose DANNAs presented in this chapter are similar to the neurons and synapses in NIDAs previously described. Neurons have a threshold parameter, and synapses have weight, refractory period, and delay parameters. In the extended design, all elements may connect to up to 16 surrounding elements, as shown in Figure 6.1.

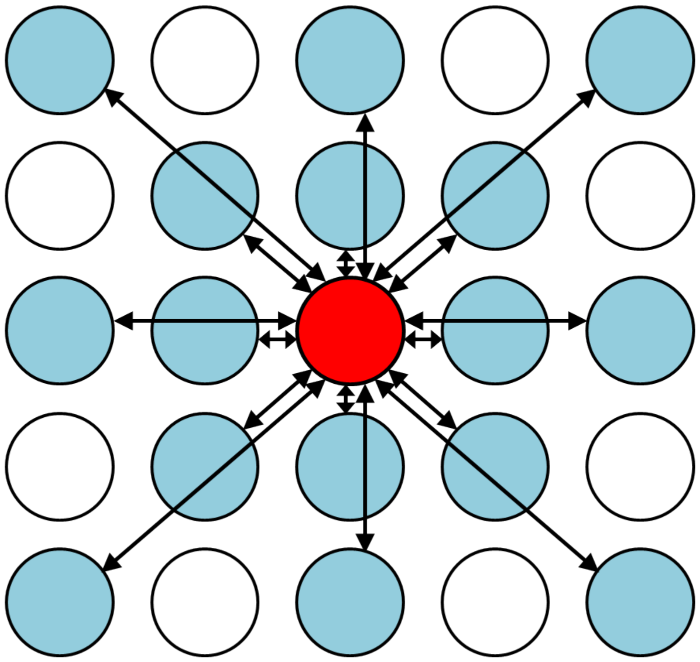


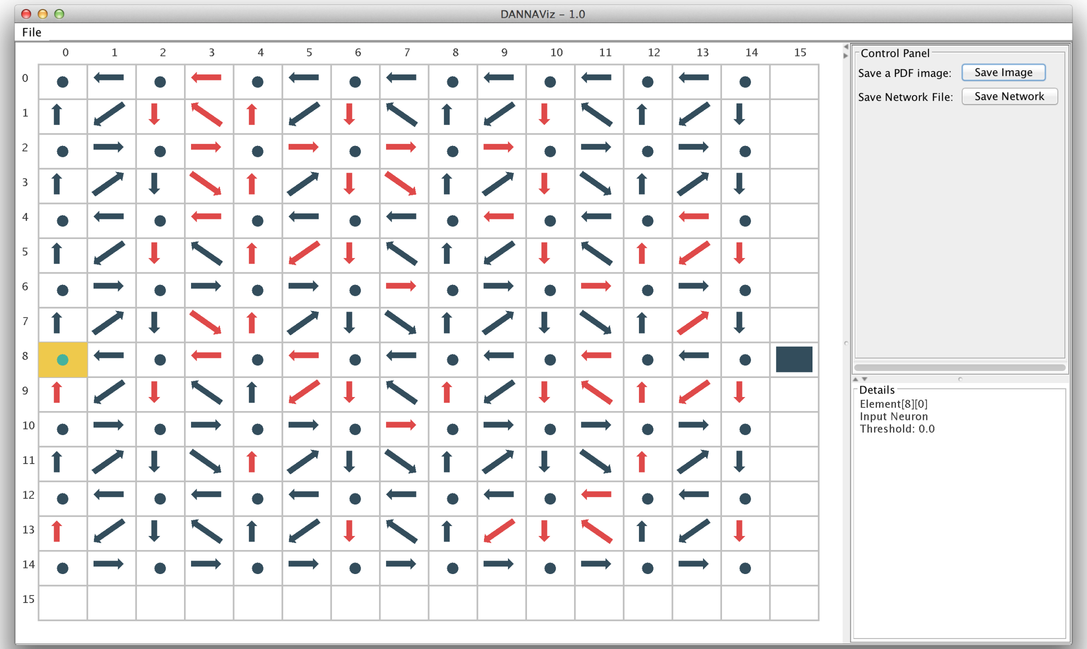
Figure 6.1: Structure of an element with its hierarchical connections. The red element connects to all of the blue elements. Every element in the array follows the red element's connection scheme. (Figure created by Jason Chan and included with his permission.)

Each of the connections shown in Figure 6.1 may be enabled as input, output, or input and output ports. When an element e fires, each of the elements connected to e through output ports, provided that its corresponding input port is enabled, receives the fire event. These elements then perform the appropriate actions for their element types.

In order to help users better understand and interact with DANNAs, we provide a graphical user interface (GUI) that includes a visual representation of the DANNAs' element types, arrangement, and connections. The GUI allows users to open, view, modify, and save DANNA networks. As we will discuss in Chapter 7, we are currently expanding the GUI to include visual analytics tools for enhanced analysis and assistance in refining the DANNA design method.

6.2 8-connection DANNA Visual Interface

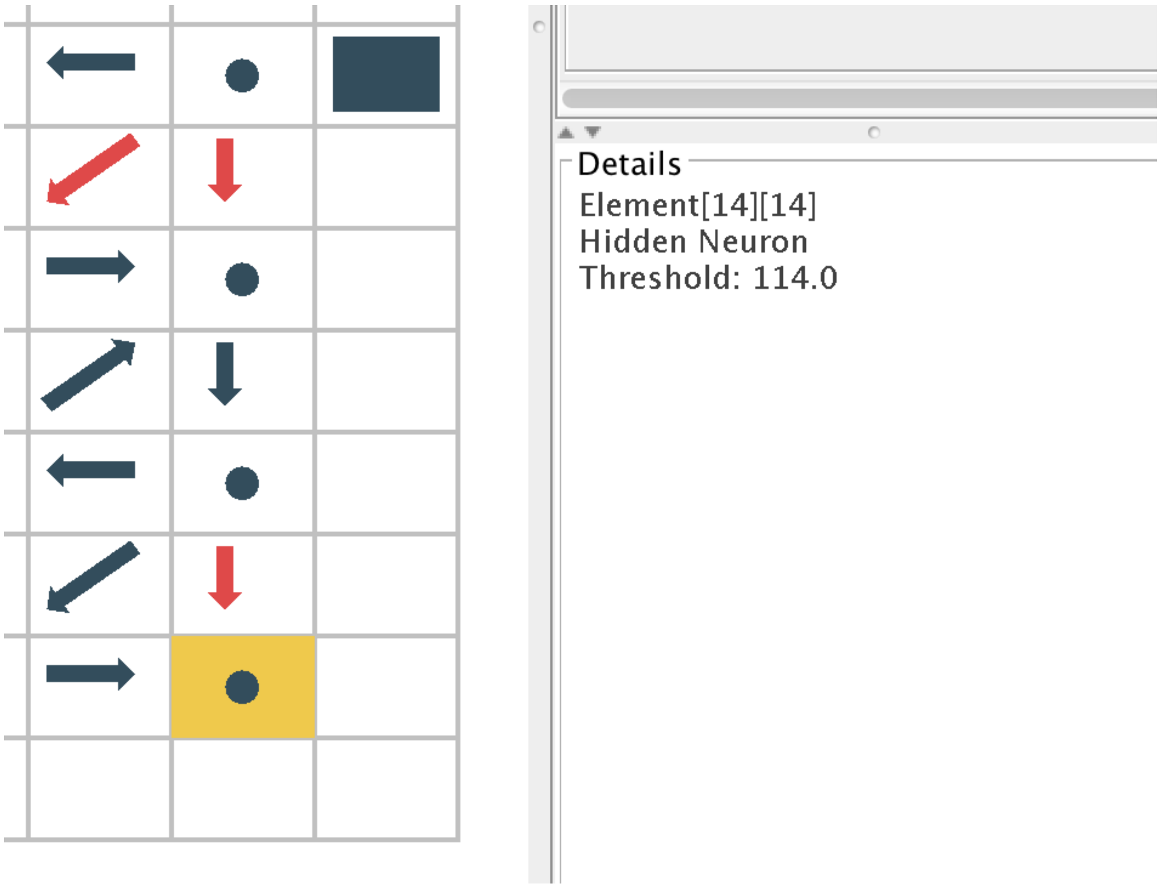
In the initial implementation of the GUI, designed for DANNAs with 8-connection elements, the user may open a network from a network file using a file chooser in the "File" menu. The default view after opening a network is shown in Figure 6.2. We call the default view of the network "grid view," since it shows the physical arrangement of elements on the DANNA hardware. Elements configured as neurons are shown as ellipses, while synapses are shown as arrows that originate from the corner of the element closest to the pre-synaptic neuron and terminate in the corner of the element closest to the post-synaptic neuron. The output synapse is shown as a solid rectangle to differentiate it from other elements. The grid view is automatically laid out to utilize the panel space fully, and it will automatically rearrange if the panel is resized. Grid boundaries and labels are drawn to indicate the overall size of the array, but elements are only drawn within the grid boxes if those elements have been configured as DANNA neurons or synapses. In the network shown, input neurons are colored teal and hidden neurons are colored navy. In this color scheme, red-colored synapses are inhibitory and navy-colored synapses are excitatory.



Element[8][0]
Input Neuron
Threshold: 0.0

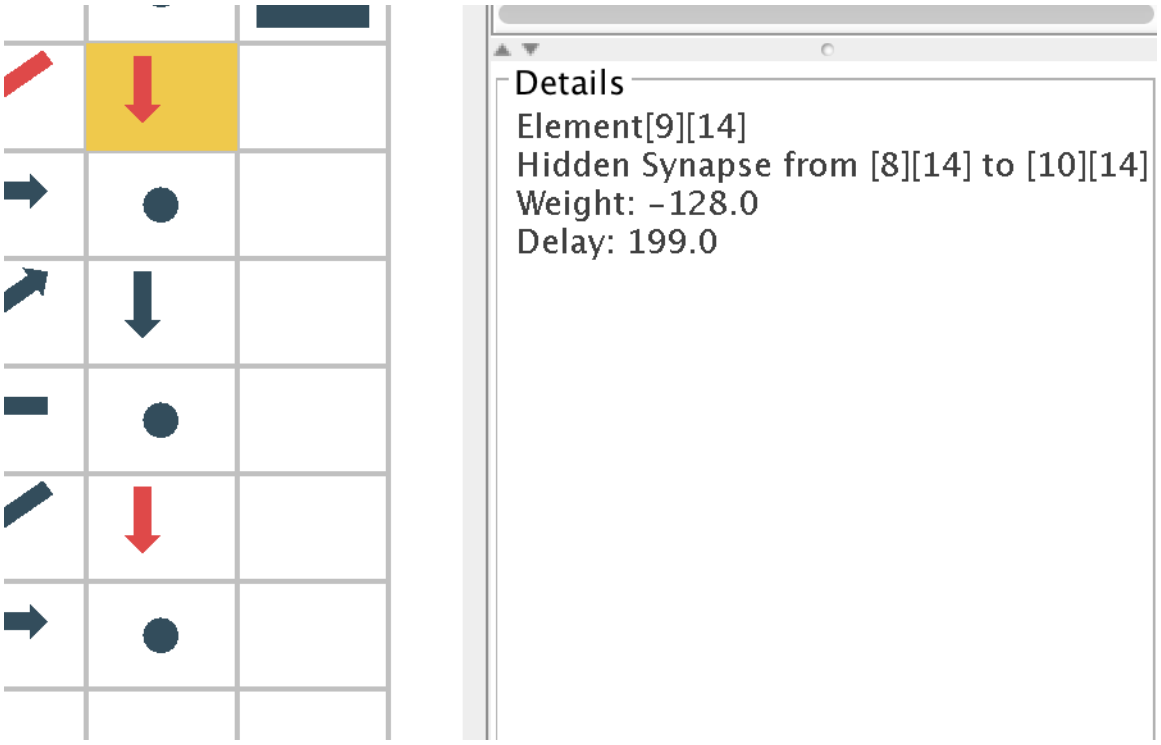
Figure 6.2: Grid view of a 16x16 DANNA network with the input neuron highlighted in yellow.

Two additional panels within the GUI assist the user in interacting with the DANNA. The control panel, shown at the top right of Figure 6.2, allows users to save the view of the network as a PDF image or save the configured network itself as a network file. The details panel, shown at the bottom right of Figure 6.2, provides a more detailed overview of highlighted elements. Elements are highlighted via mouseover in the grid view. A highlighted element has the background of its grid box colored yellow as a marker of interactivity. The details panel is linked to the grid view and displays additional information about the highlighted element. When any element is highlighted, the details panel displays the coordinates of the element as well as its status as input, hidden, or output. If a neuron is highlighted, the threshold of the neuron is also displayed. Figure 6.3 shows the highlighting of a hidden neuron and the details panel display for that highlighting. When a synapse is highlighted, the details panel displays the neurons to which the synapse connects, the synapse's weight, and its delay, in addition to the basic element properties. Figure 6.4 shows the highlighting of an inhibitory hidden synapse and the linked details panel display.



Element[14][14]
Hidden Neuron
Threshold: 114.0

Figure 6.3: Highlighting of a hidden neuron at index [14][14] and the linked Details Panel.



Element[9][14]
Hidden Synapse from [8][14] to [10][14]
Weight: -128.0
Delay: 199.0

Figure 6.4: Highlighting of a hidden synapse at index [6][7] and the linked Details Panel.

The user may configure elements within the DANNA using the GUI's right-click popup menu, shown in Figure 6.5. The user may configure the element by changing its type (neuron/synapse), by adjusting its parameters (threshold/weight/delay), by changing its pre- and/or post-synaptic neurons (for synapses only), or by removing the element from the DANNA. These interactions are coordinated through an abstract communication interface, making the GUI independent of the lower level DANNA implementation. The abstract communication interface is described in more detail by Daffron et al. [4]. If an activity file for the given DANNA has been set using a file chooser from the "File" menu, the detailed activity of selected elements may also be viewed by clicking "Display activity for selected element(s)" from the right-click popup menu.

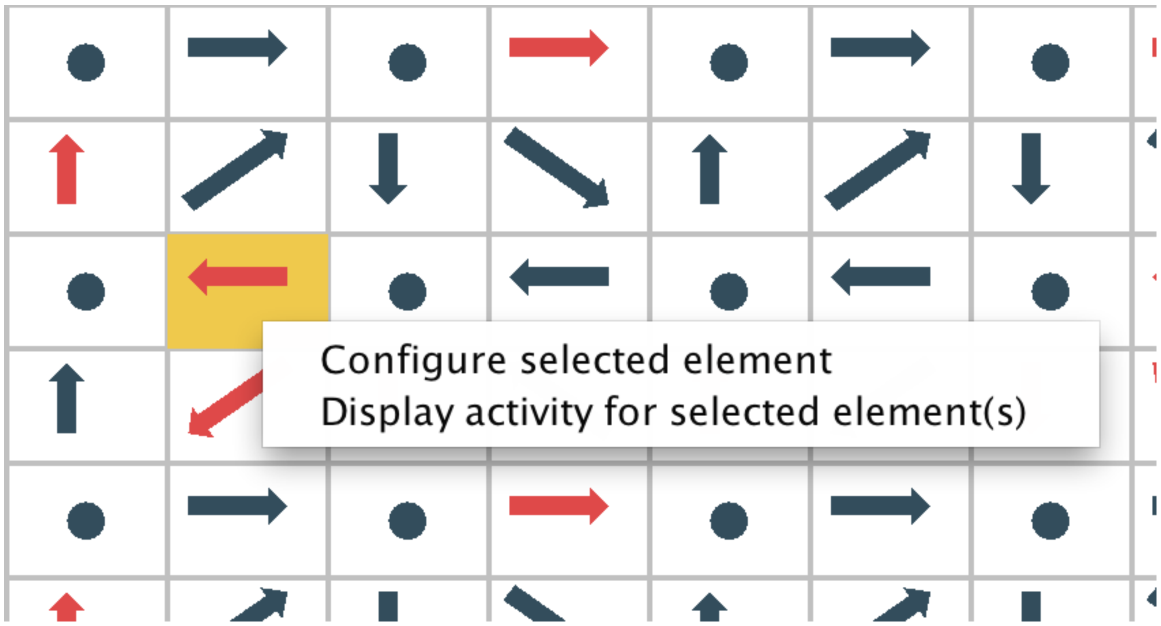
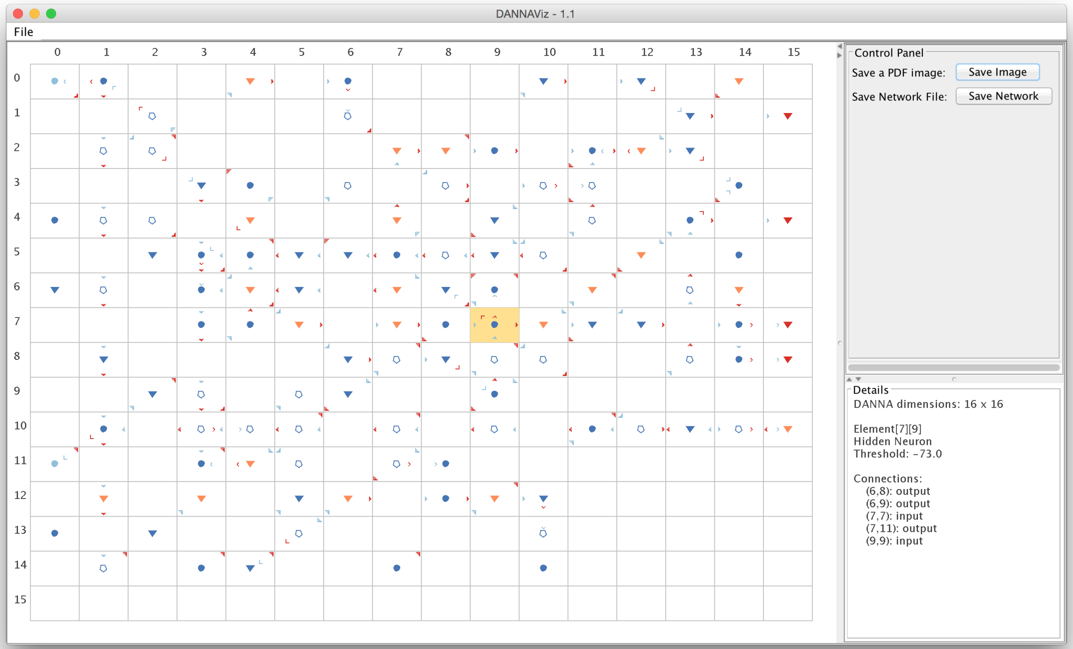

Configure selected element
Display activity for selected element(s)

Figure 6.5: Popup menu displayed upon right click of an element in the grid view.

The simple GUI provided for DANNAs can help users better understand the networks through the grid view's color and shape distinctions of elements. It also allows novice users to quickly modify and save networks without having to interact directly with the software abstraction layer. However, some of the refinements to the DANNA design that were incorporated into the 16-connection implementation could not be effectively visualized using this scheme.

6.3 16-connection DANNA Visual Interface

The visual interface for the 16-connection DANNA interface is composed of the same panels as the 8-connection interface: the grid view, control panel, and details panel. The default view for the 16-connection visual interface is shown in Figure 6.6. The grid view for this implementation is significantly more complex than that of the previous iteration in order better encode the distinctions and connections between elements for updated DANNAs. The updated DANNAs include three element types: neurons, synapses, and "passthru" elements. The functionality of neurons and synapses remains unchanged. Passthru elements are similar to synapses, and like synapses, they may have only one input connection enabled. Unlike synapses, passthru elements may have multiple output connections enabled to allow for greater fanout and connectivity of elements. The connection scheme is also more complicated in the updated DANNA implementation, so the visualization of connections will be discussed in detail in Section 6.3.3. For the updated DANNA, the configuration of the DANNA itself has been decoupled from the specific networks and elements that may be loaded onto it, so the user must open both a configuration file and a network file from the File menu to begin using the visual interface.



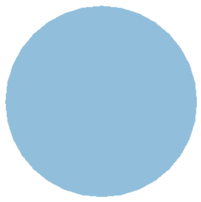
```
Details
DANNA dimensions: 16 x 16

Element[7][9]
Hidden Neuron
Threshold: -73.0

Connections:
   (6,8): output
   (6,9): output
   (7,7): input
   (7,11): output
   (9,9): input
```

Figure 6.6: Default view of 16-connection DANNA visual interface showing 16 x 16 DANNA.

6.3.1 Shape and Color Encoding for Elements

Shape is used to distinguish between element types (neuron/synapse/passthru). Figure 6.7 shows how filled ellipses represent neurons. Synapses are depicted as filled triangles of consistent shape and direction, as shown in Figure 6.8. Passthru elements are shown as pentagons, and in order to make them more distinguishable from neurons when shown at a small size, they are depicted as outlines instead of filled shapes. Passthru elements are shown in Figure 6.9.

Color is used for both elements and connections to encode node type (input/output/hidden). In the current DANNA implementation, only neurons or passthru elements may be input nodes, and only synapses may be output nodes. However, the visual interface allows for more generality, since it does not enforce these restrictions. Input nodes are light blue, as shown in Figures 6.7a and 6.9a. Hidden neuron or passthru elements are shown in a darker blue (Figures 6.7b and 6.9b). Hidden synapses are further distinguished by weight, with negative-weighted (inhibitory) synapses shown in orange and positive-weighted (excitatory) synapses shown in the darker blue, as depicted in Figures 6.8a and 6.8b, respectively. Output nodes, which are generally synapses, are colored red, as shown in Figure 6.8c. All of these colors are dark and saturated enough to show well against the yellow background highlight color. The color encodings also avoid differentiation of node type using either red and green or blue and yellow, which could be confounded by common types of color blindness.


(a) Input neuron colored light blue.

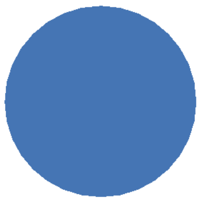
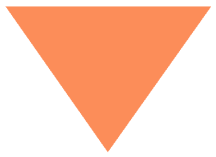
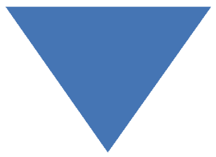

(b) Hidden neuron colored darker blue.

Figure 6.7: Neurons depicted as filled ellipses with color used to differentiate node type.


(a) Inhibitory hidden synapse colored orange.


(b) Excitatory hidden synapse colored darker blue.

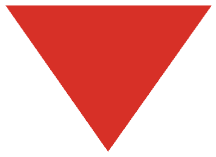
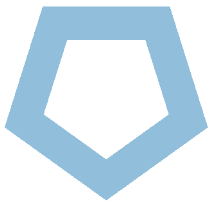

(c) Output synapse colored red.

Figure 6.8: Synapses depicted as filled triangles with color used to differentiate node type and synapse weight.


(a) Input passthru element colored light blue.

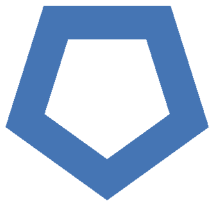

(b) Hidden passthru element colored darker blue.

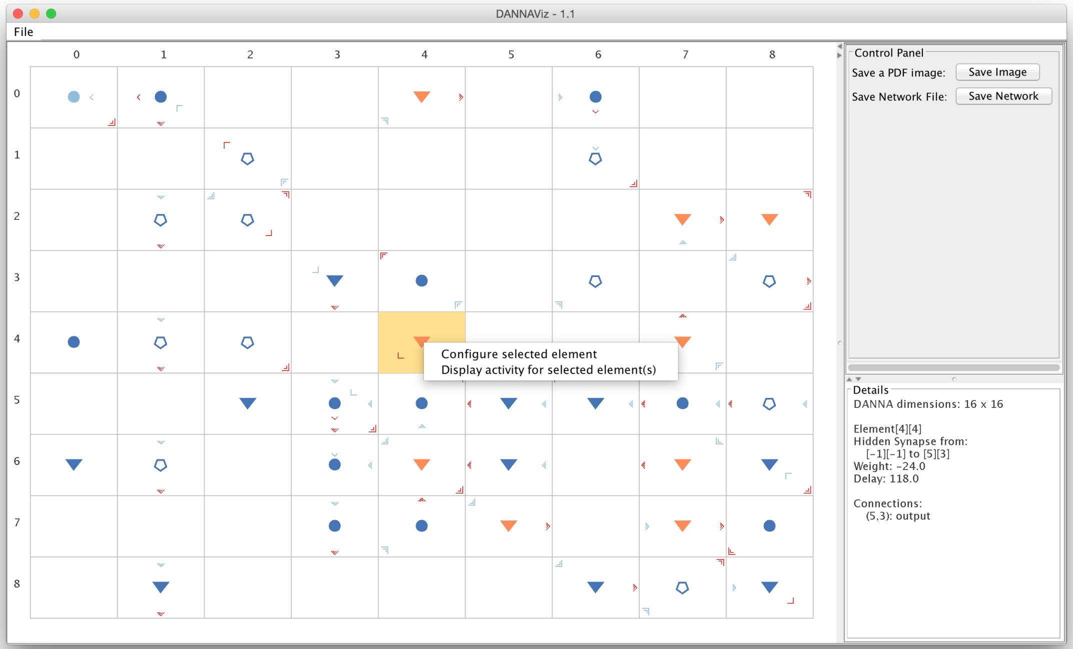
Figure 6.9: Passthru elements depicted as outlines of pentagons with color used to differentiate node type.

6.3.2 Interactions Supported

In the 16-connection visual interface, zoom, pan, and highlighting interactions are supported, as well as the popup menu and button interactions discussed in Section 6.1. The user may scroll with the mouse in either direction to zoom into the DANNA further or to zoom out. Zoom level adjusts the range of rows and columns visible, so no partial elements will be displayed upon zoom. Coordinates of elements displayed are always visible, regardless of zoom level. The maximum zoom level displays a single element, and the minimum zoom level, which is the default, displays the entire DANNA. If the DANNA is configured, its full dimensions are always shown within the details panel, so that the user can quickly see whether the entire DANNA, or only a part of it, is displayed.

Mouse drag is used to pan left or right and up or down throughout the DANNA. As with zoom, only full elements will be displayed upon pan, so mouse drag will modify the minimum and maximum rows and/or columns displayed, but will not alter the range of rows or columns visible. The mouse must be dragged approximately the current width of an element in order to change the columns displayed and approximately the height of an element to change the rows displayed. In other words, pan will move the view of the DANNA by approximately the amount dragged.

As with the 8-connection interface, when the user hovers over a particular element, highlighting will be indicated by coloring the background of the grid element yellow. Additional details about the element will be displayed within the details panel. The same buttons are available in the control panel to save an image of the grid view or the network file for the DANNA network that the grid view represents. Like in the 8-connection visual interface, a right-click popup menu allows users to configure or view activity of elements. An example of the use of the popup menu is shown in Figure 6.10. Each of these interactions may be viewed in Attachment 3 accompanying this work, which can be found in the University of Tennessee's online catalog.


DANNAViz - 1.1
File
0 1 2 3 4 5 6 7 8
Control Panel
Save a PDF image: Save Image
Save Network File: Save Network
Configure selected element
Display activity for selected element(s)

Details
DANNA dimensions: 16 x 16

Element[4][4]
Hidden Synapse from:
    [-1][-1] to [5][3]
Weight: -24.0
Delay: 118.0

Connections:
    (5,3): output

Figure 6.10: Popup menu that appears upon right-clicking an element. Menu allows for the configuration or viewing of activity of an element.

6.3.3 Visualizing Connections

In the updated DANNA implementation, each element e connects to its 8 immediate neighbors, as in the initial implementation, and e also connects to 8 elements that are two hops away, as shown in Figure 6.1. In the DANNA hardware, as well as in the abstract software layer, these connections are implemented using a hierarchy of ports in rings surrounding the element. In future implementations, the connections may be extended further in multiples of eight as additional rings of ports outside the existing rings.

The rings of connections in the 16-connection implementation are shown in Figure 6.11. The base layer of this figure, shown in lighter gray, represents the innermost ring of ports, which connects elements to their immediate neighbors. Port rings are oriented such that ports identified by the same port number are adjacent to each other (or directly diagonal to each other) on the two elements which they connect. The numbering scheme is arbitrary; other numbers or symbols can be used. The correspondence of ports with identical numbers or symbols is important, not the lettering. This is one reason symbols indicating direction and distance (one or two hops) are used in the visual interface as shown in Figure 6.10. If two elements, $e1$ and $e2$ are connected via port 0, then port 0 is oriented on both $e1$ and $e2$ so that a connection between port 0 on $e1$ and port 0 on $e2$ is the most direct path between $e1$ and $e2$. This logic requires a different arrangement of port numbers for different elements, as can be seen in Figure 6.11. The same logical arrangement of ports is extended to the second port ring, but these ports connect elements that are two hops away from each other in the DANNA grid. Example connections for the second port ring are shown in black in Figure 6.11. Note that the port numbering scheme in Figure 6.11 does not match the port numbering system in the updated DANNA implementation, but the visual interface mirrors the physical arrangement of connections in the current DANNA implementation.

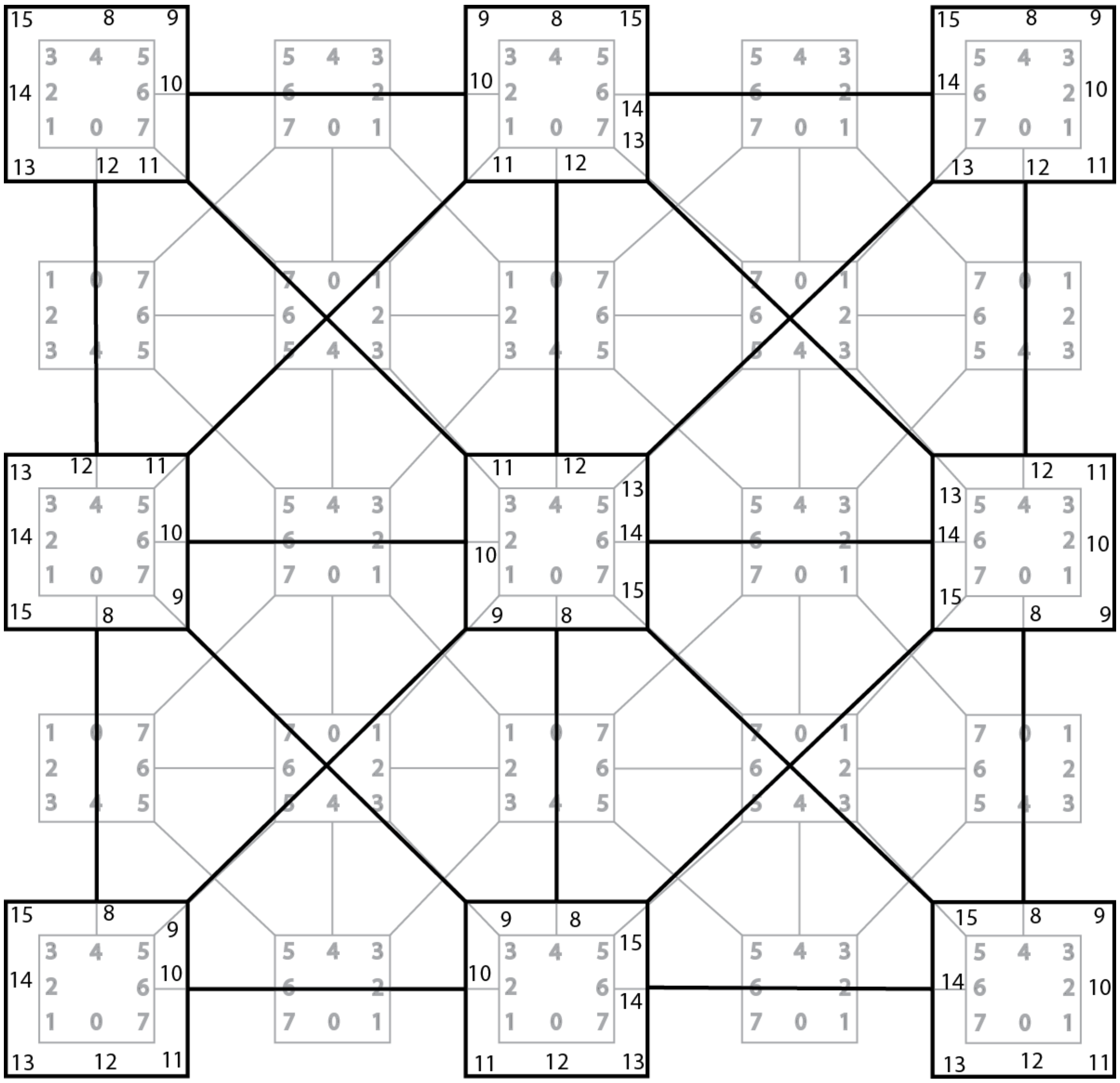


Figure 6.11: Hierarchical element connections in rings of ports surrounding elements. (Figure created by J. Douglas Birdwell and included with his permission.)

In designing the visualization scheme to represent these connections, multiple possible user goals were considered. Figure 6.12 represents a visualization whose encodings are tightly coupled with the software abstraction and hardware implementation of DANNA elements. The port rings are explicitly drawn to indicate connection level, and ports in use are encoded by color to represent input or output connection. Available ports are drawn as port numbers so that users can efficiently select desired port connections. As described previously, this labeling system for ports is arbitrary, so another set of labels may be used as long as the labels of corresponding ports match each other. A slightly modified version of this visualization is shown in Figure 6.13. The modified version adds lines of distinct stroke types to indicate direction and level of connection. These visualizations provide potentially useful information at a glance about the hardware-level implementation of DANNAs. However, ultimately it was determined that the visual clutter inherent in the design distracted too much from the information.

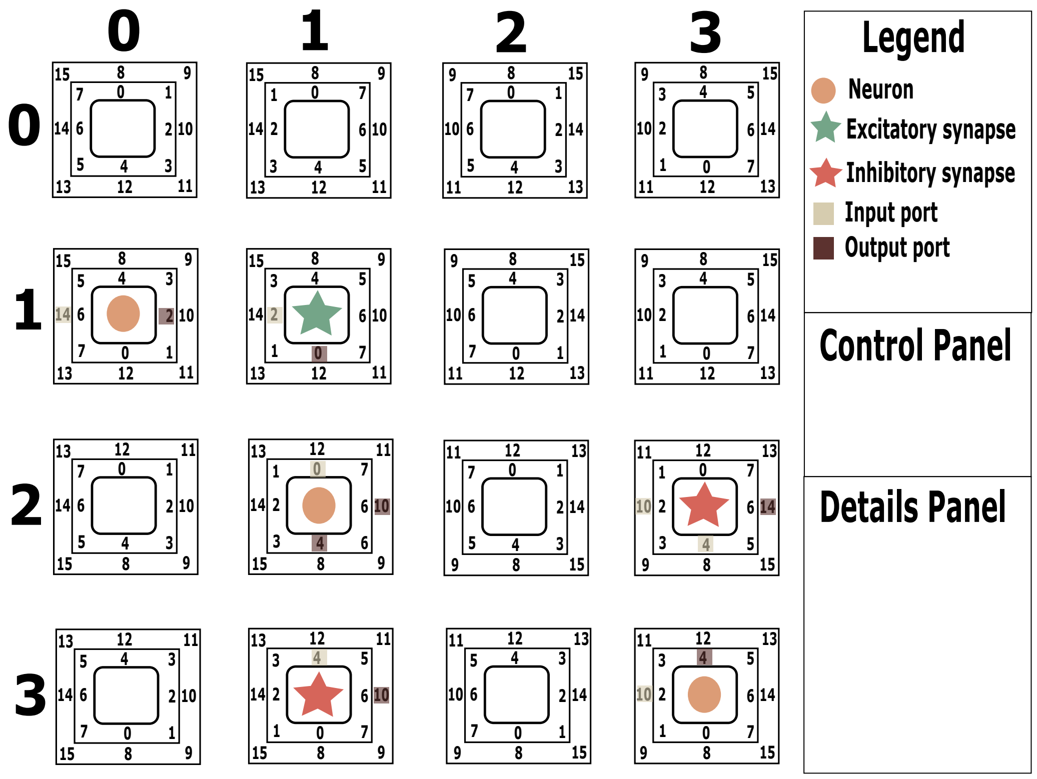


Figure 6.12: Mockup of visualization including port numbers to indicate connections.

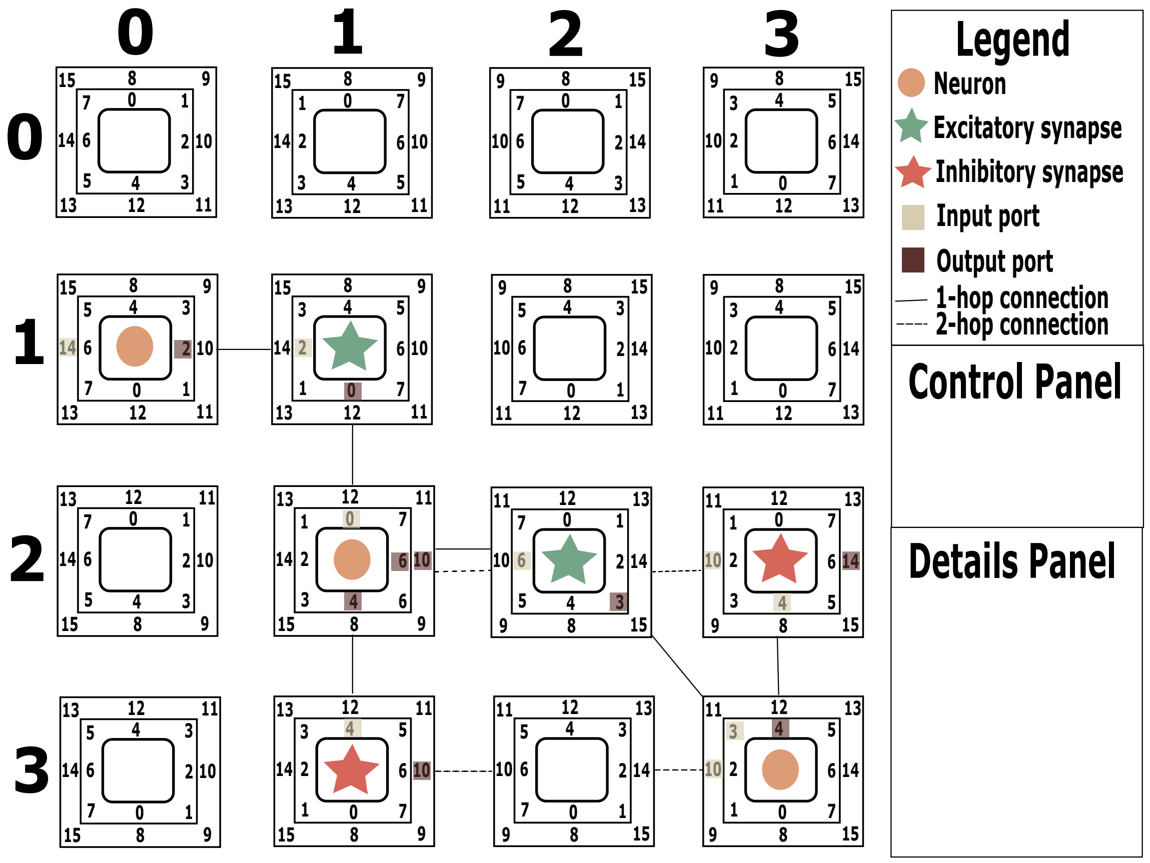
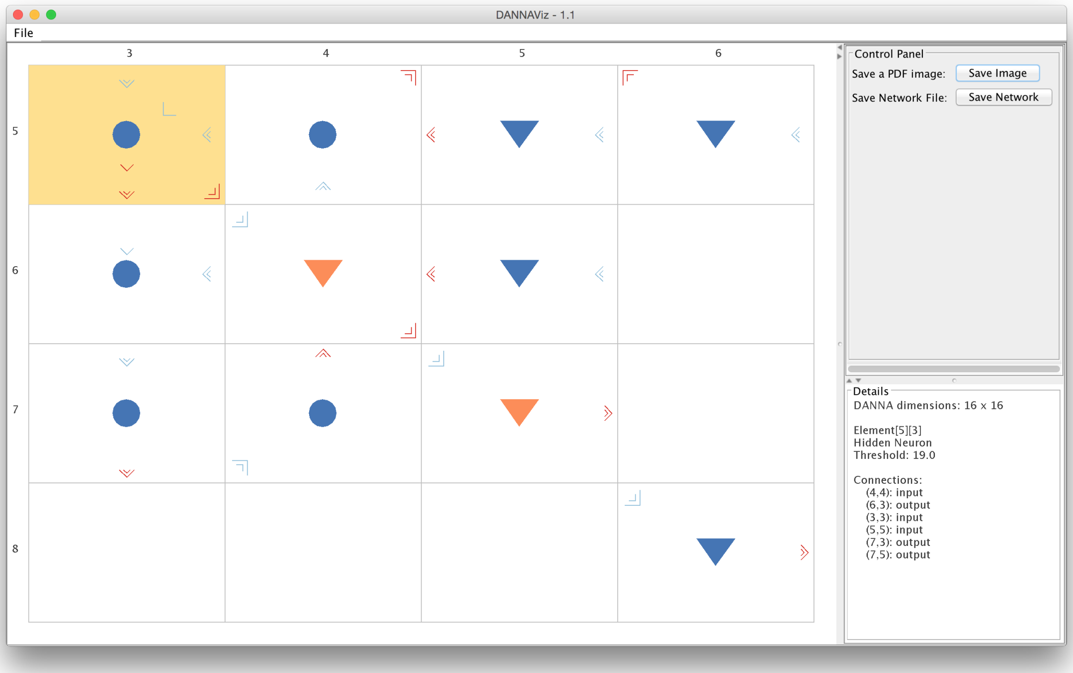


Figure 6.13: Mockup of visualization including port numbers and lines to indicate connections.

The final visualization of connections is not as explicit as the mockups shown in Figures 6.12 and 6.13, but it is cleaner and utilizes space more efficiently. Connections are depicted as arrow-like symbols instead of port numbers, and a dual encoding of color and direction indicates the element to which a connection links. The connections are still located over the appropriate ports, so 1-hop connections surround the element within an inner ring, and 2-hop connections circle the element in an outer ring. The number of marks in the symbol also indicates the level of connection, so 1-hop connection symbols contain one arrow-like mark, while 2-hop connections are composed of two nested arrow-like symbols. Similar to the lines in Figure 6.13, the arrow direction indicates the element to which a connection links. Each connection port is split into two halves, allowing for an arrow pointing inward and/or an arrow pointing outward for each connection. The inward-pointing arrows, all colored light blue, indicate that the port has been enabled as an input connection, and the outward-pointing arrows, colored red, indicate an output connection to an enabled input. The details panel has also been extended to list the detailed connections for a highlighted element. This visualization scheme allows for the representation of ports as both input and output ports, which is necessary for the updated DANNA implementation.

Figure 6.14 shows a group of connected elements. The output connection from the highlighted neuron (5,3) to synapse (7,5) is shown as the double red arrow outward in the bottom right port of the outermost ring. The corresponding connection in synapse (7,5) is depicted as the double light blue arrow inward in the top left port of the outermost ring. Note that the ring levels and number of marks for connected elements always match. The connection symbols on the connected elements are located along the most direct path between the elements. Each connection symbol is as close to the element to which it indicates connection as possible within the constraints of its port ring. Additional connections of the highlighted neuron from Figure 6.14 are shown in greater detail in Figure 6.15.

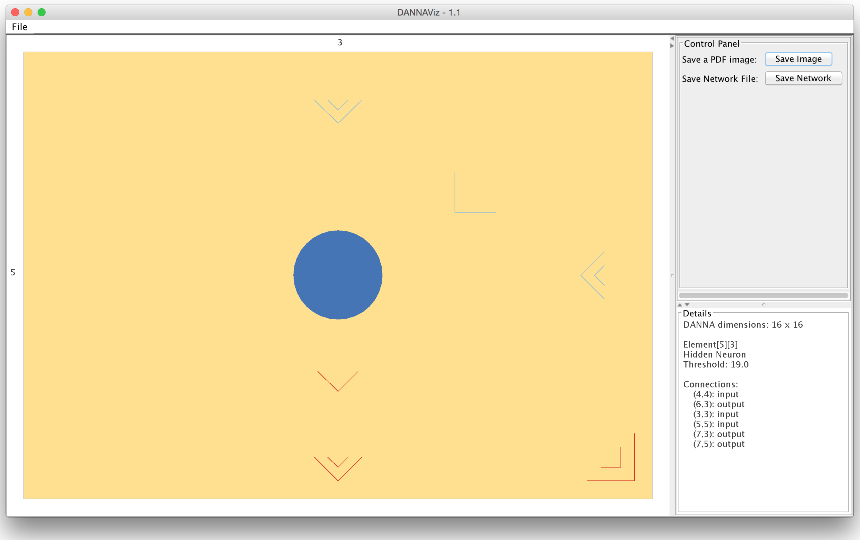


DANNA dimensions: 16 x 16

Element[5][3]
Hidden Neuron
Threshold: 19.0

Connections:
(4,4): input
(6,3): output
(3,3): input
(5,5): input
(7,3): output
(7,5): output

Figure 6.14: Zoomed-in view of a group of connected elements. Connections for highlighted element listed in details panel.



DANNA dimensions: 16 x 16

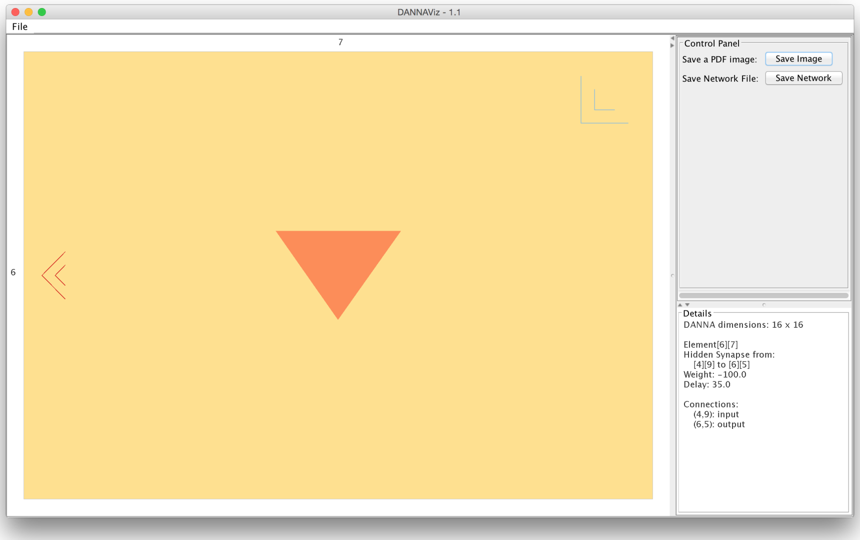
Element[5][3]
Hidden Neuron
Threshold: 19.0

Connections:
(4,4): input
(6,3): output
(3,3): input
(5,5): input
(7,3): output
(7,5): output

Figure 6.15: Zoomed-in view of a highly connected neuron, with connections listed in details panel.

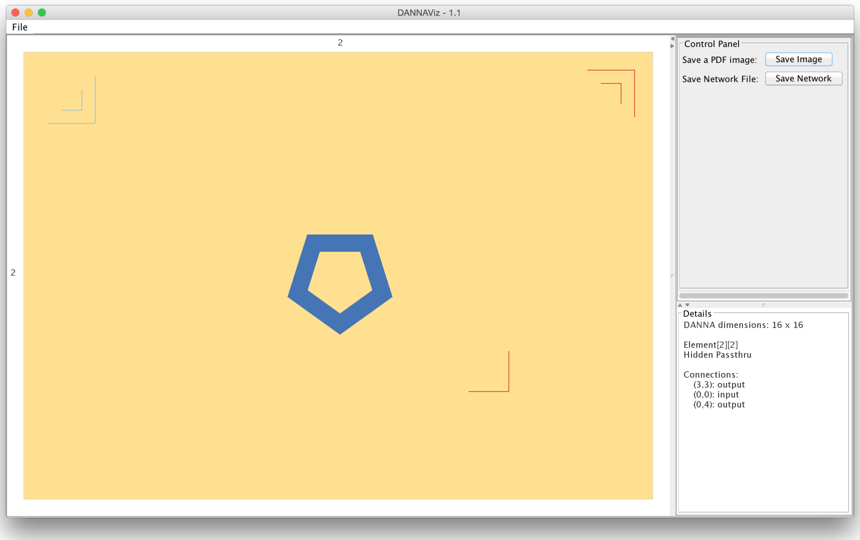
Examples of connections for synapses and passthru elements are shown in Figures 6.16 and 6.17. Synapse (6,7), depicted in Figure 6.16, has a 2-hop input connection from element (4,9) shown in the top right of the outermost ring, as well as a 2-hop output connection to element (6,5) shown in the middle left of the outermost ring. Passthru element (2,2), shown in Figure 6.17, has a 2-hop input connection from element (0,0) and a 2-hop output connection to element (0,4), both shown as double-mark arrows in the outermost ring. It also has a 1-hop output connection to element (3,3), which is shown as a single-mark arrow in the bottom right of the innermost ring.

These symbols for connections provide the same information as the mockups in Figures 6.12 and 6.13, but the symbols and color encodings have eliminated the chart junk that the mockups contained. This visualization is automatically extensible to higher levels of port rings, and the port numbers need not be displayed to indicate connectivity of elements. In future extensions to the visualization, which will be discussed in Chapter 7, connectivity will be further indicated by highlighting of the connection shapes on all elements connected to element e if e is highlighted. Links between elements will also be represented in a coordinated "network view." The network view will show the graph structure of the DANNA, with neurons represented as nodes and synapses and passthru elements represented as edges in the graph.



```
Details
DANNA dimensions: 16 x 16

Element[6][7]
Hidden Synapse from:
   [4][9] to [6][5]
Weight: -100.0
Delay: 35.0

Connections:
   (4,9): input
   (6,5): output
```

Figure 6.16: Zoomed-in view of a connected synapse, with connections listed in details panel.



```
Details
DANNA dimensions: 16 x 16

Element[2][2]
Hidden Passthru

Connections:
   (3,3): output
   (0,0): input
   (0,4): output
```

Figure 6.17: Zoomed-in view of a connected passthru element, with connections listed in details panel.

6.4 Summary

In this chapter, we have presented two iterations of the two dimensional visualization tool for DANNAs. The current implementation supports visualization of three types of elements (neurons, synapses, and passthru elements), three node types (input, hidden, or output), and up to 16 connections per element arranged in two rings of ports. The tool features capabilities for opening and configuring DANNA networks, as well as saving images or network files from them. It also supports zoom, pan, and highlighting interactions. In the next chapter, we will outline ongoing and future work to extend the three dimensional NIDA visualization and two dimensional DANNA visual interface.

Chapter 7

Future Work

7.1 Goals of Future Work

Interactivity and flexibility are the highest priorities for the visualization tools. We plan to update the NIDA tool to allow the user to modify all of the currently adjustable features from within the graphical user interface at any point during the simulation. The currently adjustable features include visibility settings as described in previous chapters, color scheme, mode of interaction (interactive vs. image rendering for video), and event selection for causality path trace. In the longer term, we intend to add additional interactivity features that allow users to explore the networks more freely. In addition to allowing visibility settings to be modified by rule (visibility upon activity, fade after inactivity, etc.), we will allow users to toggle the visibility of a selected neuron n or synapse s, along with the visibility of any other elements directly connected to n or s.

The DANNA visual interface is currently under development, and will continue to be refined in parallel with the DANNA design method and software abstraction layer. The interface will be expanded to encompass visualization of activity in addition to structure, and a coordinated network view will offer more information about the graph structure of DANNAs. Brushing and additional interactions will assist users in navigating and making sense of large DANNAs. The interface will be linked directly with the software abstraction layer for a Software Development Kit (SDK) to be made available to other researchers and new neuromorphic computing developers.

7.2 Additions to 3D NIDA Visualization

In future versions of the tool, users will also have interaction controls to define thresholds to suppress or highlight particular events. For example, the user could visualize only neurons that fire more than N times over a specified time interval, or that have fired within the last K time units. These features will allow users to eliminate visual clutter and examine critical substructures of the network in more depth. The interactivity of causality path tracing will also be extended to accommodate reverse animation in time, facilitating the exploration of causality in both directions. Further development of the visualization tool will incorporate these and other features with a focus on interactivity and the greatest flexibility possible in the exploration of networks.

Future iterations of the visualization tool will incorporate additional features for network exploration and will offer users greater interactivity to tailor the views to their own research tasks. In the short term, the visibility settings and color encodings will be expanded to give a more accurate representation of the network's current state. Specifically, in addition to the option to make network elements become visible (visibility upon activity), the tool will include a setting to reduce the visibility of elements to ghost or invisibility after a period of inactivity ("fade after inactivity"). The combination of visibility upon activity and fade after inactivity will allow users to comprehend more efficiently the propagation of activity through the network and will highlight the most active elements and substructures.

Color encodings will also be expanded in the short term to provide users with an up-to-date view of neuron charge level. Neuron hues will continue to differentiate between input, hidden, and output neurons, while saturation levels will be used to encode charge. Neuron charge level can be scaled within the range from -1.0 to 1.0, but individual neurons may have different thresholds. The visualization tool will normalize the charge level of a given neuron n with respect to the threshold of n and discretize it within a smaller number of bins (2-3). When a neuron receives charge (positive or negative) from a connected synapse, its saturation will be adjusted to the discrete level that best indicates its current proximity to the firing threshold.

We are currently expanding the functionality for the tracing of causality paths. Future versions of the tool will have the capability to trace multiple events and are expected to enable other interactions, such as control of other features of the visualization (e.g., toggling visibility of connected elements) using the event traces. Specifically, we plan to enable visualization of the causality paths with forward and backward traces through time, and we may create additional static timeline views to provide greater context for the path traces.

Additional interactivity features will be incorporated into the visualization to enhance usability and allow for greater knowledge gain from the tool. Timing of network simulations will be modifiable via YouTube-like controls within a GUI. Network direction flow will be reversible for all simulations. Highlighting of elements via mouse hover will be enabled, possibly with popup menus to view or modify more details of neurons or synapses. A mouseover tooltip may also be provided to allow for labeling or annotating of neurons or synapses. All of these features will enhance users' abilities to interact with and better understand NIDA networks.

Coordinating views may be added for specific tasks related to understanding NIDA behavior, and the default network view will be enhanced. As mentioned previously, static timeline views may be created to coordinate with causality path tracing, and they may also coordinated with a limited amount of activity in the networks in general. Activity histograms may also be provided to give users insight into the statistics of neuron and synapse behavior at a glance. Static graphs may also be provided to distinguish the activity from successful behavior and unsuccessful behavior. Unlike histograms, these graphs would retain structural information from the networks in addition to providing insight into information flow. Shading and simulated light sources can be added to improve 3D perception of the default network view. These coordinating and enhanced views will offer users greater insight into the structure and behavior of NIDA networks.

7.3 Additions to 2D DANNA Visual Interface

In the short term, the DANNA visual interface will be enhanced with coordinated views and brushing. Given the complexity of the 16-connection DANNA visual interface, a legend will be provided as one small coordinating view. More significantly, a "network view" will be added to allow for greater understanding of the graph structure of the DANNAs. The grid view is the most critical for developers who are designing and debugging DANNAs, but the network view may provide a more intuitive understanding of the network behavior. A network view will also help users better understand the NIDA architecture that DANNAs implement in hardware.

Expanded interactivity is also a priority for the DANNA visual interface. Some implementation details may also change to improve memory usage and efficiency of rendering. Highlighting on mouseover will be extended to highlight the connections (ports) of elements connected to the currently highlighted element. Additional settings may allow users to specify highlighting preferences for types of connections or paths through the DANNA from a highlighted element. In order to render more efficiently given the number of elements and connections that may need to be drawn, the base DANNA may be rendered off-screen so that only layers that have been modified will need to be re-rendered.

Since the DANNA design method, software abstraction layer, and SDK remain under active development, we expect that the users' needs for the visual interface may expand rapidly. For that reason, we have provided the foundation of the interface through the critical grid view, which is extensible to larger DANNAs with greater numbers of connections. We have also provided menus and panels to explore DANNA elements more deeply and to modify them, and these interactions are extensible to other views. In addition to incorporating visualizations of DANNA activity, we expect that other coordinating views will be necessary. We anticipate that coordinating views and visualizations may be linked effectively to the views provided in the current DANNA visual interface.

7.4 Summary

In this chapter, we have outlined some of the ongoing work and future goals for the NIDA and DANNA visualization tools. Both tools will prioritize interactivity, but will extend static coordinating views available to provide insight for specific research tasks. Extensions to the tools will be linked with the current features to offer greater insight into NIDA and DANNA structure and behavior. The next chapter provides a summary and conclusion about the contributions of this work.

Chapter 8

Conclusion

NIDA networks have been shown to solve tasks in a variety of domains, including control, anomaly detection, and classification. DANNAs, which implement NIDA networks in hardware, have also demonstrated promise as neuromorphic computational devices. In the development of a new architecture and associated design method, it can be difficult to identify which characteristics of the architecture and the method are important, as well as how to improve the overall performance of the architecture and design method. With this in mind, we are developing visualization and visual analytics tools that facilitate the understanding of both the structure of the NIDA and DANNA networks produced for different tasks and the behavior of these networks on different tasks and for different input types.

We have used the NIDA visual analytics tool presented herein to motivate analysis that can occur in real-time during the training process of the networks. For example, the hypothesis for the activity-based method for recognition of useful substructures within NIDA networks arose through exploratory interactions with the NIDA visualization and was tested as a possible extension to the NIDA evolutionary optimization method. The current visual analytics tool also features the capability to view causality paths to trace through the events that led to a particular fire or change in charge event. The NIDA design method is currently being extended to extract causality-based substructures for re-use during evolution. These causality paths provide a greater understanding to the user of the process through which networks ultimately "decide" on output.

As noted in Chapter 5, in most cases, inhibition of firing in the network is essential to the operation of the network, but it can be difficult to see the full effect of inhibition on the network's behavior without the aide of a visualization tool. The ability to see the network's full structure gives the user an intuitive feel for not only how many inhibitory synapses there are in the network, but also how active these synapses are (through highlighting of the synapse) and how many events are propagating along them (through charge points along the synapse). Extensions to the NIDA visualization tool proposed in Chapter 7 may offer further insight into the role of inhibition in network behavior.

The DANNA hardware model is being developed in parallel with the software-only NIDA networks. DANNA shares the basic features of the NIDA networks and may also be trained using evolutionary optimization. NIDA networks and their associated hardware architectures are novel, so users may not have an intuition for how the networks will behave. Moreover, hardware level errors may occur during the operation of the network, so even if users know what sort of behavior to expect from a network, they will need to see a low-level representation of what is occurring to confirm that the network is behaving as it should.

We introduced the first implementations of the DANNA visual interface in Chapter 6. The grid view and interactions provided allow users to explore the structure of a DANNA in depth, including element properties and connections. Configuration and saving interactions permit users to modify DANNA network files directly from the visual interface or use the interface to communicate with DANNAs through a software abstraction layer. Ongoing and future extensions to the DANNA visual interface, described in Chapter 7, will provide additional features to explore DANNA activity and network structure.

We are encouraged by the results that have been obtained from our NIDA and DANNA visualization tools. The tools have already improved our understanding and analysis of the structure and behavior of NIDAs and DANNAs, as well as provided a direction for improvement of the evolutionary optimization design method. We believe that further development of this tool will not only allow us to design and use NIDA and DANNA networks more efficiently, but also facilitate collaboration by providing other researchers with the means to better understand the NIDA architecture.

Bibliography

[1] Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. 26

[2] H. Bischof, A. Pinz, and W.G. Kropatsch. Visualization methods for neural networks. In *Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 581–585, Aug 1992. 27

[3] H. Bischof, W. Schneider, and AJ. Pinz. Multispectral classification of Landsat-images using neural networks. *Geoscience and Remote Sensing, IEEE Transactions on*, 30(3):482–490, May 1992. 27

[4] Christopher Daffron, Joshua C. Willis, Jason Y. Chan, Catherine D. Schuman, Margaret Drouhard, Luke Bechtel, Ryan D. Wagner, Mark E. Dean, and J. Douglas Birdwell. Dynamic adaptive neural network array of elements with hierarchical connections. Internal Report (unpublished), Department of Electrical Engineering and Computer Science, University of Tennessee, Jan 2015. 4, 77, 84

[5] Marjorie Darrah. Neural network visualization techniques. In *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*, pages 163–197. Springer US, 2006. 26

[6] Mark E. Dean, Catherine D. Schuman, and J. Douglas Birdwell. Dynamic adaptive neural network array. In Oscar H. Ibarra, Lila Kari, and Steffen Kopecki, editors, *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, pages 129–141. Springer International Publishing, 2014. 3, 26, 77

[7] Margaret Drouhard, Catherine D. Schuman, J. Douglas Birdwell, and Mark E. Dean. Visual analytics for neuroscience-inspired dynamic architectures. In

Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on, pages 106–113, Dec 2014. xi, 4, 40, 59

[8] W. Duch. Coloring black boxes: visualization of neural network decisions. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1735–1740 vol.3, July 2003. 28

[9] J.P. Eberhard, A. Wanner, and G. Wittum. Neugen: A tool for the generation of realistic morphology of cortical neurons and neural networks in 3D. *Neurocomputing*, 70(13):327 – 342, 2006. Neural Networks Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04) 7th Brazilian Symposium on Neural Networks. 29

[10] Stephen Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press, Oakland, California, 1st edition, 2009. 6, 7, 8, 10, 14, 16, 18

[11] C.J. Figueroa and P.A Estevez. A new visualization scheme for self-organizing neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, pages –762, July 2004. 28

[12] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008. 26

[13] Padraig Gleeson, Volker Steuber, and R. Angus Silver. Neuroconstruct: A tool for modeling networks in 3D space. neuron 54:219235, 2007. 29

[14] Jeff Hawkins and Dileep George. Hierarchical temporal memory: Concepts, theory and terminology. *Whitepaper, Numenta Inc*, 2006. 26

[15] Christopher G. Healey, Laura Tateosian, James T. Enns, and Mark Remple. Perceptually based brush strokes for nonphotorealistic visualization. *ACM Trans. Graph.*, 23(1):64–96, 2004. 8, 9, 10, 11, 18, 21

[16] Geoffrey E. Hinton, Terrence J. Sejnowski, and David H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA, 1984. 26

[17] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001. 28

[18] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits, 2009. `http://yann.lecun.com/exdb/mnist/`. 4, 38

[19] B. R. Linnell. Quadrant-distance graphs: a method for visualizing neural network weight spaces. In *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, volume 3, pages 1666–1671 vol.3, 1999. 27, 28

[20] Henry Markram. The blue brain project. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06. ACM, New York, NY, USA, 2006. 25

[21] Henry Markram, Karlheinz Meier, Thomas Lippert, Sten Grillner, Richard Frackowiak, Stanislas Dehaene, Alois Knoll, Haim Sompolinsky, Kris Verstreken, Javier DeFelipe, Seth Grant, Jean-Pierre Changeux, and Alois Saria. Introducing the human brain project. *Procedia Computer Science*, 7(0):39 – 42, 2011. Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11). 26

[22] Dharmendra S. Modha, Rajagopal Ananthanarayanan, Steven K. Esser, Anthony Ndirango, Anthony J. Sherbondy, and Raghavendra Singh. Cognitive computing. *Commun. ACM*, 54(8):62–71, August 2011. 26

[23] Casey Reas, Ben Fry, and John Maeda. *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, 2007. 41

[24] Ronald A. Rensink. Change detection. *Annual Review of Psychology*, 53(1):245–277, 2002. PMID: 11752486. 10, 11, 14, 21, 22

[25] G. Romero, P.A. Castillo, J.J. Merelo, and A. Prieto. Using SOM for neural network visualization. In Jos Mira and Alberto Prieto, editors, *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, volume 2084 of *Lecture Notes in Computer Science*, pages 629–636. Springer Berlin Heidelberg, 2001. 28

[26] Catherine D. Schuman and J. Douglas Birdwell. Dynamic artificial neural networks with affective systems. *PLoS ONE*, 8(11):e80455, 11 2013. 1, 25, 32

[27] Catherine D. Schuman and J. Douglas Birdwell. Variable structure dynamic artificial neural networks. *Biologically Inspired Cognitive Architectures*, 6(0):126 – 130, 2013. 1, 25

[28] Catherine D. Schuman, J. Douglas Birdwell, and Mark E. Dean. Neuroscience-inspired inspired dynamic architectures. In *Biomedical Science and Engineering Center Conference (BSEC), 2014 Annual Oak Ridge National Laboratory*, pages 1–4, May 2014. 1, 2, 53

[29] Catherine D. Schuman, J. Douglas Birdwell, and Mark E. Dean. Spatiotemporal classification using neuroscience-inspired dynamic architectures. *Procedia Computer Science*, 41:89–97, 2014. xi, 1, 25, 38, 39, 40

[30] Catherine Dorothy Schuman. *Neuroscience-Inspired Dynamic Architectures.* PhD thesis, University of Tennessee, May 2015. 1, 39

[31] Nina Thiessen. Connection maps: A new way to visualize similarity relationships. Master's thesis, University of Toronto, 2004. 27

[32] Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut, 2nd edition, 2001. 5, 7, 11, 12, 13, 14, 16

[33] F.-Y. Tzeng and Kwan-Liu Ma. Opening the black box - data driven visualization of neural networks. In *Visualization, 2005. VIS 05. IEEE*, pages 383–390, Oct 2005. 28

[34] M. Uzak, I. Vertal, R. Jaksa, and P. Sincak. Reduction of visual information in neural network learning process visualization. In *Applied Machine Intelligence and Informatics, 2008. SAMI 2008. 6th International Symposium on*, pages 279–284, Jan 2008. 28

[35] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map, 2000. 28

[36] C. Ware. *Information Visualization: Perception for Design*. Interactive Technologies. Elsevier Science, 2012. 7, 9, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21

[37] T.A. Yang. Development of a distributed visualization system for neural network simulation. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, volume 4, pages 2873–2876 vol.4, 2001. 28

[38] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423 –1447, sep 1999. 26

[39] Hujun Yin. Visom - a novel method for multivariate data projection and structure visualization. *Trans. Neur. Netw.*, 13(1):237–243, January 2002. 28

[40] B. Zanetti, A.N. Ide, and J.H. Saito. A framework for neural networks simulation and visualization - neocognitron case. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–485–8, Sept 2005. 28

Appendix

Appendix A

Videos Included with this Work

A.1 NIDA Visualization Videos

Attachment 1. Non-interactive Video of NIDA Visualization: Video-Mode-NIDAViz.mp4

Attachment 2. Interactive Video of NIDA Visualization: Interactive-Mode-NIDAViz.mp4

Since the NIDA visualization is a three dimensional tool, the attached videos help demonstrate its capabilities much more effectively than static images. The videos also demonstrate the two modes: video-mode and interactive, and they exhibit the interactions available in interactive mode. The color scheme for the videos is discussed in Section 5.3.

Attachment 1 provides a narrated visualization of a NIDA network trained using evolutionary optimization to recognize the digit zero in handwritten digit images from the MNIST database. This video combines images rendered in video mode in the NIDA visualization tool with input images from MNIST data set and a graph of output firing events. The left panel shows the scanning of the input images (0, 5, and 2) across columns, the central panel shows the network processing the input image, and the right panel shows the graph of output firing events.

Attachment 2 shows the NIDA visualization tool in use on a network trained to recognize the handwritten digit 7. The mouse is used to start the simulation of activity via the timing buttons at the top left. Mouse drag rotates the network about any axes, and mouse scroll is used to zoom in and out. The default visibility mode of the network is ghost, and visibility upon activity is used to render elements fully visible after they have been active during the simulation.

A.2 DANNA Visualization Videos

Attachment 3. Interactive Video of DANNA Visual Interface: InteractiveDANNAViz.mp4

The attached video of the DANNA visual interface, Attachment 3, demonstrates standard usage and interactions supported by the tool. The interactions are described in greater detail in Section 6.3.2. In the video, the user selects a configuration file, "config.json" via the "Open Configuration File" option in the File menu. The user then selects a specific DANNA file, "64rand.json" to be loaded via the "Open Network File" option in the File menu. Once the DANNA has been loaded, mouse scroll is used to zoom in and out of the network, and mouse drag is used to pan left or right and up or down. Details for highlighted elements are shown in the details panel at the bottom right of the interface. Upon right-click, the configuration popup menu is shown.

Vita

Margaret (Meg) Drouhard earned her Bachelor of Arts in German Studies and Spanish Studies from American University in 2009. Her undergraduate research focus was transitional justice and recovery from gross human rights violations. In 2009, Meg was awarded a Fulbright Teaching Assistantship Fellowship for Germany and an Austrian-American Educational Commission Teaching Assistantship Fellowship for Austria. After studying and working abroad for a few years in Chile, Germany, and Austria, Meg joined the University of Tennessee, and she graduated with her Master of Science in Computer Science in May 2015. Her master's research is in the field of visualization and visual analytics.