NEUROMORPHIC MODELS OF THE AMYGDALA WITH APPLICATIONS TO SPIKE BASED COMPUTING AND ROBOTICS

by

Kate D. Fischl

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

May, 2019

Abstract

Computational neural simulations do not match the functionality and operation of the brain processes they attempt to model. This gap exists due to both our incomplete understanding of brain function and the technological limitations of computers. Moreover, given that the shrinking of transistors has reached its physical limit, fundamentally different computer paradigms are needed to help bridge this gap. Neuromorphic hardware technologies attempt to abstract the form of brain function to provide a computational solution post-Moore's Law, and neuromorphic algorithms provide software frameworks to increase biological plausibility within neural models. This thesis focuses on utilizing neuromorphic frameworks to better understand how the brain processes social and emotional stimuli. It describes the creation of a spiking-neuron computational model of the amygdala, the brain region behind our social interactions, and the simulation of the model using brain-inspired computer hardware, as well as the implementations of other spike-based computations on these hardwares. Although scientists agree that the amygdala is the main component of the social brain, few models exist to explain amygdala function beyond "fight or flight".

ABSTRACT

This model incorporates neuroscientists' more nuanced understanding of the amygdala, and is validated by comparing the neural responses measured from the model to responses measured in primate amygdalae under the same experimental conditions. This model will inform future physiological experiments, which will generate deeper neuroscientific insights, which will in turn allow for better neural models. Repeated iteratively, this positive feedback loop in which better models beget better understanding of biology and vice versa will help close the gap between the computer and the brain. The computer networks and hardware that emerge from this process have the potential to achieve higher computing efficiency, approaching or perhaps surpassing the efficiency of the human brain; provide the foundation for new approaches to artificial intelligence and machine learning within a spike-based computing paradigm; and widen our understanding of brain function.

Primary Reader: Andreas G. Andreou

Secondary Reader: Ralph Etienne-Cummings

Committee Members: Sridevi Sarma and Philippe Pouliquen

Acknowledgments

There are so many people who collaborated with me throughout this research, for whom I am incredibly grateful.

I would like to thank my advisor, Andreas Andreou, for providing me with many opportunities throughout this program. From him I have learned the importance of scientific communication and the importance of understanding scientific impact. Through his support, I met and collaborated with many other scientists. I am thankful for all of this.

I would like to thank Ralph Eitenne-Cummings, Sridevi Sarma and Philippe Pouliquen for being members of my committee and their support and guidance over the course of my PhD.

I would like to thank the past and present members of the Andreou Lab who I overlapped with during the course of my PhD including Tom Murray, Dan Mendat, Tomas Figliolia, Guillaume Garreau, Kayode Sanni, Gaspar Tognetti, Martin Villemur, Chris Sapsanis, Alejandro Pasciaroni, Jeff Craley, Valerie Rennoll, Jonah Sengupta, and Pedro Julian. Specifically I would like to thank Jeff Craley for cre-

ating much of the code that the Bayesian changepoint detection work was based off of, and being a great partner for many course assignments and projects. I would like to thank Dan Mendat for all of his help setting up and maintaining the TrueNorth ecosystem in our lab, his collaboration on TrueNorth, as well as his general support and kindness.

I would like to thank the members of the Computational Sensory-Motor Systems Laboratory for being great co-members of the 4th floor of Barton. Specifically I thank Adam Cellon for his work on the amygdala social robot (Amy G. Dala) and his friendship. I would also like to thank Jack Zhang for driving to and from Telluride with me in 2014, and offering such helpful advice along the way, as well as letting us detour over an hour to visit the four corners.

I am so thankful for the scientific guidance, collaboration, and support of Kati Gothard and Terry Stewart. I thank Terry for the frequent skype meetings, for answering all of my long emails, and his inspirational love of cognitive science. I thank Kati for sharing her love of exploring processing in the brain and specifically the amygdala. Her thirst for knowledge was an inspiration. I greatly enjoyed all of our interactions and am thankful for the opportunities working with her provided me.

I would like to thank Prisca Zimmerman, Philip Putnam, JJ Morrow, and Clayton Mosher (members of the Gothard Lab) for sharing their data with me and answering many questions about it along the way. I would like to thank Sebastien Ballesta and Jeno Szep for their collaboration on the work analyzing eyeblinks.

ACKNOWLEDGMENTS

In addition to Terry Stewart, I would like to thank all members of the Nengo team and the staff of the 2018 Nengo Summer School, but specifically Chris Eliasmith and Aaron Voelker for their general assistance at the Telluride 2018 Neuromorphic Cognition Engineering Workshop in helping me run the simplified amygdala Nengo models on different neuromorphic hardware.

I would like to thank the entire IBM TrueNorth team for all of their help and assistance over the course of my research with TrueNorth. I would specifically like to thank Andrew Cassidy, Guillaume Garreau, Ben Shaw, Paul Merolla, John Arthur, Rodrigo Alvarez-Icaza for their help at the Telluride 2015 and 2016 Neuromorphic Cognition Engineering Workshops, the 2015 TrueNorth Bootcamp, and their general support throughout my PhD work.

I would like to thank the members of the Loihi team who attended the Telluride 2018 Neuromorphic Cognition Engineering Workshop and assisted me in running my Nengo amygdala model on Loihi, specifically Mike Davies, Jon Tse, and Andreas Wild. Similarly I would like to thank the Braindrop team who attend the Telluride 2018 Neuromorphic Cognition Engineering Workshop and the 2018 Nengo Summer School who assisted me in running my Nengo amygdala model on Braindrop, specifically Alex Neckar, Sam Fok, Nick Oza, and Kwabena Boahen.

I would like to thank Wei-Yu (Will) Tsai and Jack Sampson for their collaboration on the path planning work.

I would like to thank Timothy Horiuchi for his collaboration in building the socio-

emotional robot.

I would like to thank Garrick Orchard for his collaboration on work utilizing TrueNorth, and his general insight on spike-based computing and pursing a PhD at JHU.

I would also like to thank all JHU ECE and CLSP staff who booked rooms, coordinated GBOs, and did many many other logistical tasks for me during the course of my PhD, specifically Janel Johnson, Ruth Scally, Nicole Aaron, Cora Mayenschein, Debbie Race, Dana Walter-Shock, Yamese Diggs, and Melissa Gibbins. I am very grateful for all that you have done and continue to do.

I would like to thank the National Science foundation for awarding me the NSF graduate research fellowship under Grant No. DGE-1232825. It gave me the freedom to pursue much of this research and collobrate with all of these individuals. I would also like to thank the NSF Grant 1540916: SL-CN Cortical Architectures for Robust Adaptive Perception and Action through the Telluride Workshop on Neuromorphic Cognition Engineering. Attending the Telluride workshop each summer was instrumental to this work, and I am grateful that the organizers allowed me to return so many times.

On a personal level, I want to thank the member of GRACE (the Graduate Association [of women] in CS and ECE). I am so grateful for the community we have built and hope it continues to enrich the experience of women in CS and ECE. I am thankful to my friend Huda Khayrallah for forming GRACE with me, and being a

huge support during this program. I would also like to thank Rachel Rudinger for her friendship and support. Additionally I am also grateful to have had Jane Foster, Alycen Wiacek, Arlene Chiu, Michelle Graham, and Stephanie Graceffo in the ECE department with me and as friends throughout this process.

I am also thankful for my Baltimore friend family for all of the onesie bike rides, themed costumes, dance dance resolutions, and Robot Eta videos during my time in Baltimore. I am specifically grateful for the friendship of Sarah Edelsburg, Elisheva Goldwasser, Eta Flamholz, Liz Imber, Rachel McGrain, Rachel Bergstein, and Sydnee Chavis. I am specifically thankful for Rachel Bergstein's "cow cow" motto that I adopted during the final year of this work.

I am thankful for the support of my friends from college, specifically Rebecca Kaufman, Avital Ludomirsky, Bethy Atkins, Sarah Lux, Josh Oppenheimer, Alex Rosen, Karen Azani, Fiona Maguire, and Alex Satty. I look forward to many more orange weekends at reunions.

I am also thankful for my friendship with Jacob Kreimer, and all of the female scientist postcards he mailed me throughout this program.

I am incredibly grateful for Brian Rayburn. The happiness and balance you have brought me in the final months of this process have been wonderful. I look forward to our future adventures.

My success during this process is in large part to the strong female friendships I have found. I am so thankful to have met Kaitlin Fair at my first Telluride workshop.

ACKNOWLEDGMENTS

I am so grateful for her scientific collaborations, but also so lucky to have her in my corner as a friend. Without her friendship, I do not think this thesis would have become a reality. I am also thankful for our co-admiration of the music of Taylor Swift which provided an inspirational background to shake it off and complete much of this work.

I am also so thankful for my family. I am sadden that my grandfather, Robert Fischl, will never read this thesis, but am honored to place it on my bookshelf next to his and my father's, Dan Fischl's, theses.[1,2] I have been inspired by his love of research and scientific achievements in the face of many challenges throughout his life. I am thankful for all of my grandparents and their support and love, as well as that of my aunts, uncles, and cousins. I am thankful for my brother, Alex Fischl, and his girlfriend, Sigal Middleton. It is lovely to have family I love and enjoy being with. I am thankful that my parents, Lee Anne Fischl and Dan Fischl, encouraged me to pursue engineering and have always supported me. I am grateful to be their daughter, and am thankful for all they have enabled me to achieve.

Lastly I am thankful for Rebecca, Brian, Eta, Adam, Dan, Bert, Kaitlin, and Rachel who helped proofread this work.

Dedication

This thesis is dedicated to the women who came before me and the women who will come after me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the creation of the first digital computer, the ENIAC,[3] in 1946, the field of computer processing has witnessed a complete transformation. The ENIAC weighed almost 50 tons, occupied 1,800 square feet, and used 18,000 vacuum tubes. Today many semiconductor companies create integrated circuits using processes less than 20nm in size. In 2017, Apple released the iPhone 8, containing an Apple A11 Bionic processor. The A11 was manufactured using a 10nm FinFET process and contained 4.3 billion transistors. For the last 30 years, the decrease in transistor size has followed Moore's Law,[4] a "law" coined by Gordan Moore in 1965 to characterize the increase in the number of transistors per chip over time. As the number of transistors per chip has grown and transistor size has continued to shrink, semiconductor companies have come up against the fundamental limits of physics. In 2016, researchers from Lawrence Berkeley National Laboratory created a molybdenum disulfide (MoS2) transistor with

a physical gate length of 1 nm that used a single-walled carbon nanotube as the gate electrode.[5] This work effectively side-stepped the prediction that silicon (Si) transistors fail below 5 nm gate lengths due to short-channel effects. Nonetheless, engineers, scientists, and researchers alike agree that Moore's Law has come to an end: Moore is no more.[6]

Despite the end of Moore's Law, the combination of faster processing, Internet of thing (IoT) devices, and a prevalence of real-time processing have resulted in an explosion of data and advances for the fields of machine learning, artificial intelligence, and robotics.[7] It is estimated that 2.5 quintillion bytes of data are generated every day and that 90% of the data in the world was generated in the last two years (as of 2018).[8] This sharp increase in data and machine learning research coupled with the end of Moore's Law begs the question, how will computer technology be able to keep pace with the emergence of these new processing demands?

One path forward post-Moore's law lies in the field of neuromorphic engineering. Neuromorphic engineering creates technology solutions that emulate the brain, the most efficient "computer", with the hopes of producing lower-power, faster computer processors.[9–11] Carver Mead first coined the term "neuromorphic" in 1989,[9] when he linked transistor properties to those of a biological neuron and first proposed creating silicon chips to mimic biological structures and computation. The human brain performs many different types of complicated tasks while only burning about 20% of the whole body's energy budget, despite composing only 2% of the body's total mass.[12]

This far exceeds any efficiency realized by modern computers. Although computers may be faster at electrical conduction, the brain is optimized for information storage and representation to a degree of efficiency we are still unable to understand. Traditional Von Neumann computer architectures generally process information in a sequential fashion, with memory and computing resources often occupying different physical spaces within a processor. This configuration expends energy and time to shuttle information to and from memory for computations. Neuromorphic processors capitalize on the massively parallel nature of the brain, which consists of many parallel "neurons" or computation units that co-locate memory and computational resources to avoid some of the pitfalls of traditional processors. To date there have been neuromorphic processors designed under a variety of goals and constraints, implementing a wide range of computational neuron models.[11,13–20]

Like many computing-adjacent fields, neuroscience has been affected by the increase in processing power linked to Moore's Law and has seen vast technological advancements over the last half century. In 1952, Alan Hodgkin and Andrew Huxley published the first in a series of papers on their groundbreaking work in understanding how the flow of ions within neurons generates action potentials, allowing neurons to send electrical signals throughout the body.[21] Hodgkin and Huxley proposed a simple circuit model to describe ion flow based on their work using the patch-clamp method to measure the current-voltage relationship in the membrane of the squid giant axon. Over the next seven decades, neuroscience experimentation exploded

resulting in a vast development of neural recording technology.[22–24] Developments in recording technology have greatly increased the number of neurons that can be simultaneously recorded, increased the quality of the signal recorded, decreased the size and impact of recording probes, and changed the way in which recordings are made. Stevenson and Kording have quantified this trend by describing a "Moore's Law" of neuroscience.[25] They predict based on past trends that in 15 years ($\sim$ 2025), scientists will be able to record simultaneously from 1,000 neurons. This growth in neural data fundamentally changes the questions scientists can pose and investigate through experimental research. With increased amounts of neural data, the need for new analysis methods and algorithms grows.

This thesis focuses on the intersection of neuromorphic engineering and neuroscience. It contains two main contributions, one scientific and one technological. Its scientific contribution explores neuromorphic modeling techniques to better understand how the brain processes social and emotional stimuli. This is done by creating a neuromorphic computational model of the amygdala, the region of the brain responsible for processing social and emotional stimuli. All of the data analyzed for this modeling work came from a collaboration with neuroscientists in Professor Katalin Gothard's neurophysiology group at the University of Arizona. The model was validated using primate amygdala single unit neuron recordings and goes beyond existing work by incorporating multiple emotional states instead of focusing only on a fear conditioning paradigm. The model also takes into account individual nuclei and

their contribution to the amygdala's overall function, a distinction lacking in existing amygdala computational models..

This thesis's technological contribution lies in understanding and then leveraging the gains and limitations of current neuromorphic processors for neuroscience modeling, spike-based computing, and robotics. In 2014 IBM announced their TrueNorth Neurosynaptic System, the first industry-produced neuromorphic processor.[13] With one million neurons, TrueNorth presented a unique opportunity to develop large-scale, spike-based algorithms and realize many of the theoretical gains of neuromorphic processing. This thesis describes multiple spike-based algorithms developed on TrueNorth, as well as a project that utilized the TrueNorth ecosystem to execute a trained convolutional neural network on hardware as a part of an autonomous platform. Additionally, it compares a number of neuromorphic processors including Intel's Loihi,[14] University of Manchester's SpiNNaker,[16,26,27] and Stanford University's Braindrop[15] to understand the trade-offs and benefits of these processors for modeling and spike-based computation.

This thesis begins in Chapter 2 by introducing the amygdala and its role in social and emotional processing. A number of statistical methods and data analysis techniques were used to better understand socio-emotional processing in the primate amygdala. Their methodology and results are explained in this chapter. In particular, the chapter discusses the role of eyeblinks as a social determinant, and then explores changes in neural firing patterns in response to the eyeblinks. Further detail

is presented through the implementation of a Bayesian changepoint detection algorithm for detecting neurons that exhibited a change in firing rate as a response to the eyeblinks. Chapter 2 also explains research conducted using point process models to predict the firing of individual neurons as a result of measured covariates such as input stimuli, past spiking history, and the behavior of nearby neurons. Chapter 2 concludes by describing a tool used to separate multiple bat echolocation calls when flying in groups. This tool supported research for understanding social behaviors seen in bats, specifically changes in echolocation patterns when they fly in groups. Overall chapter 2 focuses on statistical methods and data analysis techniques to better understand behavioral and neural responses to socio-emotional processing.

Chapter 3 of this thesis describes models the amygdala at a higher level of abstraction, utilizing population-level modeling for understanding how the amygdala processes social and emotional stimuli. This chapter details a computational spiking-neuron model of the amygdala built using the Neural Engineering Framework.[28,29] This model goes beyond existing models by incorporating multiple emotional states rather than focusing solely on a fear response. It also includes a breakdown of nuclei functionality instead of modeling the amygdala as one homogeneous structure as most previously existing models do. To validate the model, its neurons' responses were compared with single unit neuron recordings from the primate amygdalae. There was a high degree of matching between the distribution of responses measured from the neurons in the model to those recorded from neurons found in the primate amygdalae.

Chapter 4 shifts to the technological contributions of this thesis by focusing on research by the author utilizing the neuromorphic processor, TrueNorth. This chapter gives an overview of the TrueNorth architecture and then details work accomplished using TrueNorth in each of its two engineering methodologies. The first methodology uses IBM's energy-efficient deep neuromorphic networks (Eedn)[30] framework to train convolutional neural networks to execute on TrueNorth directly. The second methodology uses the Corelet Programming Environment (CPE)[31] to develop circuit configurations, or corelets, to execute natively on TrueNorth and perform specific computations. This chapter discusses applications on TrueNorth in the areas of spike-based computation and robotics.

Lastly, Chapter 5 explores additional neuromorphic hardware beyond the TrueNorth chip. Intel's Loihi,[14] University of Manchester's SpiNNaker,[16,26,27] and Stanford University's Braindrop[15] are analyzed to understand the trade-offs and benefits of each of these processors. Practical considerations when using each of these hardwares are discussed, as well as the results of benchmark experiments. Additionally the chapter details the corelets constructed to execute the NEF on TrueNorth, enabling its comparison amongst these other neuromorphic processors.

This work presents research both within the field of computational neuroscience as well as computer engineering. Given the technological advances in computer hardware, computer processing, neural experimentation, data analysis, and machine learning, the intertwined progress of these fields becomes even more apparent. More data

requires more powerful processing, which in turn poses more interesting and complicated scientific questions which require more data. This work not only presents a computational model of the amygdala, a brain region not well understood, but also analyzes the role of computation in neuroscience itself. As we enter into an era beyond Moore's Law, alternative processing methodologies will become paramount to progress in other fields.

Chapter 2

Statistical Methods and Data Analysis to Understand Socio-Emotional Processing in the Amygdala

How the brain processes social and emotional stimuli is not well understand. Although scientists agree that the amygdala is the main component of the social brain, the amygdala lies deep in the brain which makes it a difficult region to reach and record. Because the amygdala is highly connected to other parts of the brain,[32,33] it is difficult to isolate its inputs, thus complicating experimental setups and the analysis of functional behavior. The amygdala's primary role involves analyzing social

interactions and contributing to stimulus appraisal, relevance detection, activation of neuroendocrine response, and somatic motor expressions of emotions.[34] These brain activities form the foundation of emotional responses.

The amygdala is an almond shaped structure located in the anterior part of the medial temporal lobe in the limbic system.[32,33] Figure 2.1 shows a simplified representation of the processing flow through the amygdala. Sensory inputs arrive to the lateral (input) nucleus from cortical association areas, such as the superior temporal sulcus, posterior parietal, and inferotemporal cortices.[35,36] In the lateral nucleus, stimuli are identified and discriminated. Output from the lateral nucleus then converges with inputs from the orbitofrontal and medial prefrontal cortices in the basal nucleus.[36,37] Prefrontal areas likely signal to the amygdala the current value or behavioral significance of a stimulus or event.[33,38–41] As such, the basal nucleus is where the identity ("what is it") of the stimulus is combined with its significance ("what does it mean"). Large pyramidal cells in the basal nucleus provide feedback projections that signal the outcome of these computations to multiple cortical areas.[42,43] The next stage of processing takes place in the accessory basal nucleus, which is less well understood and, as the name suggests, might only be an extension (or duplication) of the basal nucleus.[44] The basal and accessory basal nuclei both project to the central nucleus. The central nucleus is bi-directionally connected to autonomic centers in the brain stem and hypothalamus.[45] Consequently, activity in the central nucleus is thought to trigger autonomic and behavioral responses to stimuli of high emotional

value ("what to do").[39,46,47] This thesis focuses on six main nuclei: the lateral nucleus, basal nucleus, accessory basal nucleus, central nucleus, media nucleus, and the anterior amygdaloid area (AAA), although parts of this work combine nuclei to further simplify the subdivisions within the amygdala while modeling.

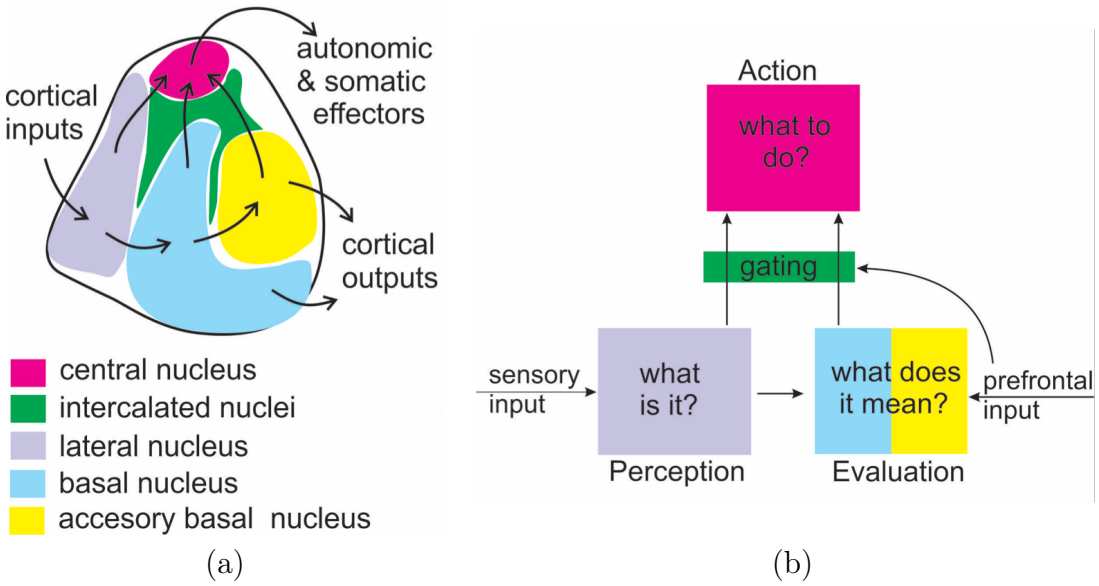


Figure 2.1: The flow of information into, within, and out of the amygdala. (a) Five major nuclei are shown and (b) their corresponding role.

The amygdalae of monkeys are often considered when studying social behavior because of their hierarchical, social societies,[48,49] and because the monkey amygdala has been shown to be a reasonable proxy for the human amygdala.[50] Although the amygdala of humans resembles that of monkeys, there are differences in the overall size and nuclei breakdown between the two species. The human brain is estimated to contain 86 billion neurons, with 12 million neurons within the amygdala. The monkey brain is estimated to contain 6.4 billion neurons, with 1.7 million neurons within

the amygdala. Although the human brain is $\sim$ 10 times larger than the monkey brain and the human amygdala is $\sim$ 7 times larger than the monkey amygdala, the relative scaling of the different amygdala nuclei does not follow the scaling of the overall structure. Table 2.1 lists the number of neurons per amygdala nuclei in humans and monkeys and Table 2.2 lists the volume of amygdala nuclei in humans and monkeys.[51–53] The values for the monkey amygdala are valid for both males and females, as well as both the left and right amygdalae.[53] As will be detailed in the next section, all of the neural data analyzed in this thesis comes from amygdalae of adult male rhesus macaques (Macaca mulatta).

	Human	Monkey
Brain	86×10^9	6.38×10^9
Amygdala	12.21×10^6	1.7×10^6
Lateral	4×10^6	1.59×10^6
Basal	3.24×10^6	1.25×10^6
Accessory Basal	1.28×10^6	0.89×10^6
Central	0.36×10^6	0.30×10^6
Medial	-	0.28×10^6
Paralaminar	-	0.41×10^6
Other	3.33×10^6	-

Table 2.1: Average number of neurons in the brain,[54,55] amygdala, and amygdala nuclei for humans[51] and monkeys[53] .

There are many neural and physiological responses that result from the amygdala's emotional processing.[33,40,56–58] In this chapter the role of eyeblinks as a social indicator is analyzed,[59] but humans and animals exhibit many physiological signals to indicate underlying emotional states.[58,60–63] Non-human primates, which is what the work in this thesis will focus on, communicate many socio-emotional states through

	Human[51]	Monkey[53]
Brain	-	52,360
Amygdala	44.54	192.60
Lateral	14.84	38.40
Basal	11.53	47.15
Accessory Basal	4.94	24.38
Central	1.08	8.15
Medial	-	5.42
Paralaminar	-	8.84
Remaining	12.26	-

Table 2.2: Average volume (mm^3) of the brain, amygdala, and amygdala nuclei in humans and monkeys.[51–53]

facial expressions, eye contact, and grooming.[60,64,65] These responses are more easily measured than neural recordings, and if correctly linked to specific socio-emotional processing could lead to more straightforward experimental paradigms for studying the social brain.

This chapter describes different statistical methods and data analysis techniques to better understand socio-emotional processing in the primate amygdala. These methods use different statistical methodologies to understand and ask questions from the collected data. They focus primarily on the analysis of physiological and single neuron responses to social experiments. Subsequently, Chapter 3 describes a nuclei-level modeling approach to understanding socio-emotional processing in the amygdala, and discusses a subdivision of labor between the nuclei of the amygdala. Creating effective models of amygdala socio-emotional processing will not only increase our understanding of this brain region, but by linking underlying neural processes to measurable psychological responses, it provides an avenue to measure, monitor, and

understand socio-emotional processing in the human brain.

2.1 Data

Much of the work described in this chapter is a collaboration with Professor Katalin Gothard's group at the University of Arizona. All of the neural and behavioral data analyzed here came from adult male rhesus macaques (Macaca mulatta), and was collected by members of her group. The rhesus brain organization and structure is similar to that of humans. Rhesus monkeys share many social behaviors with humans. Rhesus monkeys live in large, hierarchical social groups. When monkeys first meet, they engage in a series of negotiations, involving both aggressive and friendly facial expressions, eye contact, gestures, and posture stances to determine which monkey will emerge with a higher rank.[60,64,65] These visual signals provide a rich framework to supplement the neural recordings and provide real-time data on the emerging social interaction.[66,67] Analysis of these responses provides invaluable insight into human social interactions because monkeys and humans share both analogous social interactions and closely related neural architecture. Models of these socio-emotional neural and behavioral responses enable a greater understanding of the related brain areas.

The work described here analyzes data collected over the course of ten years in Professor Katalin Gothard's group. All experiments were performed in compliance

with National Institute of Health guidelines for the use of primates in research and were approved by the Institutional Animal Care and Use Committee at the University of Arizona. Although the data was not collected originally for any of the analyses described in this thesis, it was obtained to better understand the amygdala and its processing, and ask related scientific questions.

In addition to the single unit neuron recordings analyzed as a part of this work, electromyography (EMG), eye tracking data, pupil size, videos of the monkey, and the stimuli videos or images were recorded. Neuron nuclei was determined during post processing. Data analyzed in this thesis comes from one of two types of experiments: either the monkey was shown socially relevant images[57] or the monkey was shown socially relevant movies.[68] Spikes were sorted and processed from the single unit neuron recordings prior to any analysis described in this thesis.

2.2 Social Determinants of Eyeblinks in Adult Male Macaques

Blinking serves multiple purposes *. The reflexive closure of the eyelids maintains the moisture of the cornea and protects the eyes from foreign objects.[69–72] The rate and timing of the eyeblinks, however, does not merely reflect the physiological status of the eyes. In both humans and non-human primates, blinking has been linked to

*This section was previously published by the author.[59]

cognitive states and to social engagement with conspecifics.[73–79] Eyeblinks play a role in social communication. Indeed, humans often attribute mental states to their social partners based on observed changes in their blinking behavior.[80–82] Furthermore, humans coordinate the timing of their blinks with the blinks of their social partners.[83,84] This phenomenon, called eyeblink entrainment, is absent when the social partners are prevented from fully engaging with each other (e.g., seeing each other speak without any audio to convey the message).[83–85] Such observations suggest that eyeblink entrainment is not an automatic imitation of blinking but an elemental form of social interaction.

Macaque monkeys may also entrain their eyeblinks to one another during real-life dyadic social interactions. It is unknown whether videos of natural social behaviors, constructed to serve as a proxy for dyadic social interactions, can also induce eyeblink entrainment, or other social behaviors in viewer monkeys. Previous studies have shown that videos with social content induce several interactive social behaviors, such as gaze following, the reciprocation of eye contact and facial expressions.[68,86–88] Videos depicting social stimuli, however, cannot fully substitute for real-life interactions because they are limited by a major shortcoming: the behavior of the stimulus monkey remains unchanged despite the viewer's attempt to respond to the perceived social signals and engage the protagonist. Nevertheless, videos are valuable stimuli for neurophysiological studies because they can be presented multiple times and their presentation can be coupled with both non-invasive physiological monitoring (e.g., eye

tracking, autonomic recordings) and invasive measures of brain activity (intracranial recordings). If the ultimate goal is to understand the neural events that govern social behavior in primates, it is critical to use the most appropriate stimuli to elicit mental states in laboratory settings that closely resemble the mental states in real-life dyadic interactions.

The aim of this work was to determine the social factors that predict when monkeys blink while they view videos of natural social behaviors displayed by unfamiliar conspecifics. Based on previous observations that monkeys display natural social behaviors toward monkeys shown in videos,[68,86–89] as though they are attempting to socially engage them, we hypothesized that the blinking behavior in response to videos would be comparable to blinking behavior during real-life social interactions. We predicted that monkeys would entrain their eyeblinks while watching videos, just as humans entrain their eyeblinks during real life social interactions. We expected that their blink frequency would be modulated by the emotional expressions of their social partners.

2.2.1 Results

Four male monkeys, QT, RI, RU, and ZI viewed 178, 130, 143 and 330 unique ten second long videos respectively over at total of 62 recording sessions (QT=16 sessions, RI=10 sessions, RU=13 sessions, ZI=23 sessions). The majority of the videos depicted unfamiliar monkeys (henceforth stimulus monkeys), placed in a plexiglass cage where

they displayed socially meaningful facial expressions, postures, and gestures. Most of these videos depicted only one monkey, but a subset of these videos showed two or more monkeys (13% of video exposures). We also displayed videos of individual monkeys or groups of monkeys in natural outdoor settings. These video segments were recorded in the field station of the California National Primate Research Center and on the field station of Cayo Santiago. The segments were not explicitly chosen to show facial expressions, but on occasion facial expressions are visible. Of the 367 videos, 99 videos clips were seen by all four monkeys. Each monkey viewed each video 3-15 times.

Eyeblink rate decreased significantly when the monkeys watched the videos (Wilcoxon sum rank test, on averages/session, QT: $p = 1.86\times10^{-6}$, RI: $p = 0.011$, RU: $p = 0.040$, ZI: $p = 1.84 \times 10^{-5}$; Figure 2.2a, compared to baseline period when monkeys viewed a blank screen). The reduction in blink rate correlated with the content of the videos (Figure 2.2b). Videos depicting more than one monkey or monkeys in outdoor environments induced a larger decrease in eyeblink rate than videos depicting a single monkey in an indoor environment (Wilcoxon sum rank test, $p = 4.37 \times 10^{-8}$, (one monkey indoors vs. one monkey outdoors); $p = 1.76 \times 10^{-4}$ (one monkey indoors vs, multiple monkeys indoors), and $p = 3.10 \times 10^{-19}$ (one monkey indoors vs. multiple monkeys outdoors) (Figure 2.2b). We observed no significant differences among movies that depicted more than one monkey indoors, one monkey outdoors, or multiple monkeys outdoors (Wilcoxon sum rank test, one monkey outdoors vs. multiple

monkeys indoors: $p = 0.70$, one monkey outdoors vs. multiple monkeys outdoors: p=0.30 and multiple monkey indoors vs. multiple monkeys outdoors $p = 0.71$) (Figure 2.2b).

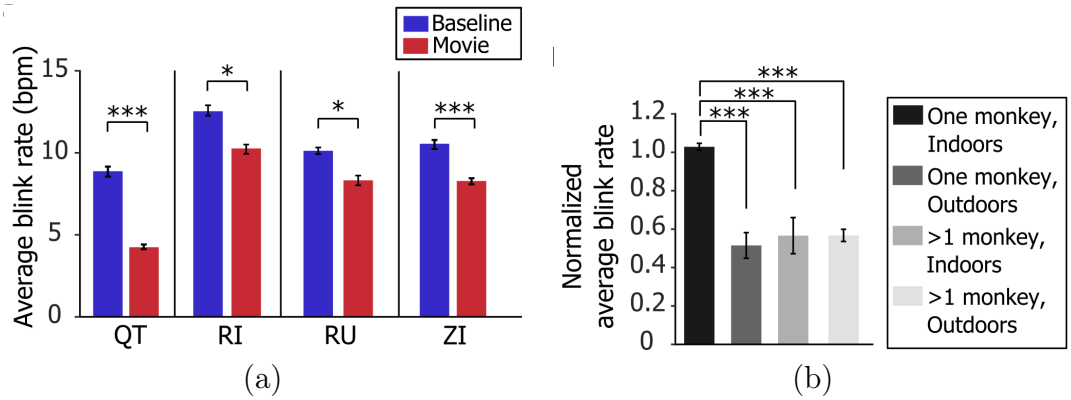


Figure 2.2: Video watching lowered the blinking rate of the viewer monkeys. (a) Average +SEM blink rate (blinks per minute = bpm) during video viewing (red bars) compared to baseline (blue bars). Each of the four monkeys blinked significantly less when viewing movies (Wilcoxon sum rank test, QT: $p = 1.86 \times 10^{-6}$, RI: $p = 0.011$, RU: $p = 0.040$, ZI: $p = 1.84 \times 10^{-5}$). (b) Eyeblink rate depends on the social complexity of the movie content. Eyeblink rates have been normalized to the average blink rate during video viewing of each monkey. The blinking rates during videos of different content were compared using a Wilcoxon two-tailed rank sum test. Asterisks indicate significant differences. Eyeblink rate decreased significantly during videos that occurred in natural settings and videos that depicted multiple monkeys, $p = 4.37 \times 10^{-8}$, $p = 1.76 \times 10^{-4}$, $p = 3.10 \times 10^{-19}$, respectively as graphically displayed. For both (a) and (b) $^*p < 0.05$, $^{***}p < 0.001$, Wilcoxon rank sum test. Error bars represent SEM.

We next explored how the occurrence of viewers' eyeblinks correlated with the unfolding of the stimulus monkeys' socio-emotional behaviors. We found that, even though the viewer monkey's blink rate was reduced during video viewing compared to baseline, the eyeblinks appeared synchronous at particular moments during the viewings. The blinking of the viewers clustered across multiple viewings of the same

video (Figure 2.4), suggesting that blink rate is related to the visual and/or socio-emotional content of the videos. This clustering appeared both for repeated viewings by the same monkey and across monkeys. Indeed, the probability of blinking in a window of 400 ms ($\pm$200 ms from the blink in a different viewing) was higher than chance (ANOVA, $p < 0.05$; Figure 2.4). This synchronization of blinking across trials and among monkeys is unlikely to be due to low-level visual features. Indeed, the probability of blinking was not significantly correlated with the amount of motion in the videos (quantified at pixel-by-pixel changes in brightness) (Spearman rank correlation, QT, $R = -0.035$ ($p = 0.44$), ZI, $R = -0.061$ ($p = 0.18$), RU, $R = -0.050$ ($p = 0.27$), RI, $R = 0.021$ ($p = 0.64$). Rather, the increases in blink synchrony appear to be the result of socio-emotional factors in the videos.

To identify the specific behavioral events that might cause the clustering of eye-blinks, we explored the relationship between the social signals emitted by the stimulus monkey (gaze direction, facial expression) and blinking behavior of the viewer monkey. We found that the viewer monkeys blinked more frequently when the stimulus monkey displayed a facial expression directed at the viewer monkey (permutation test, $p < 0.05$, for the specific of this test, see methods; Figure 2.5). All four monkeys showed a tendency to blink more frequently when the stimulus monkey's gaze was directed at the viewer. However, this increase in blink rate depended on the facial expression of the stimulus monkey. Three of the four monkeys blinked more often while looking at the direct gaze of a stimulus monkey with a threatening facial ex-

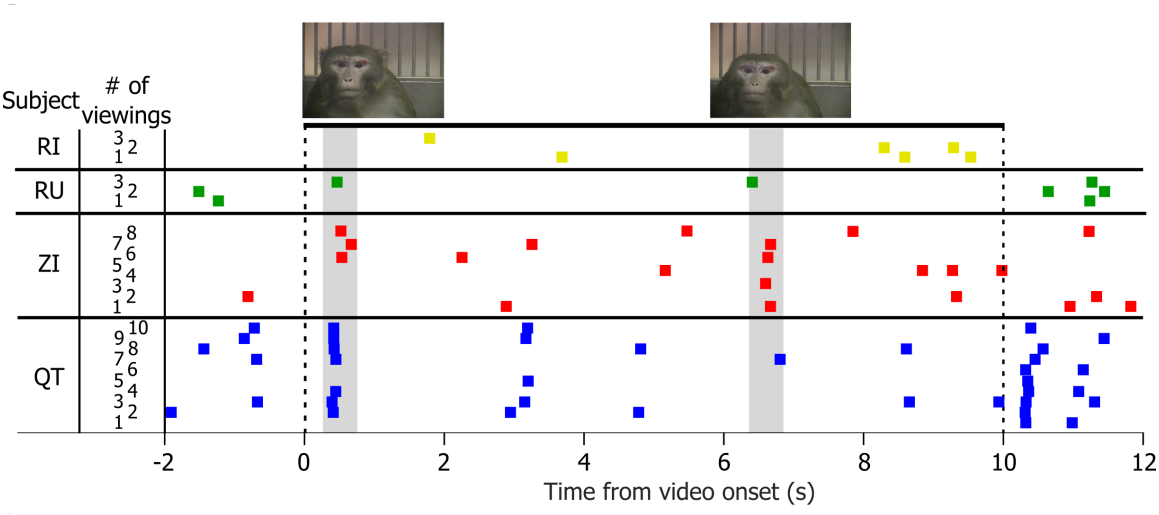


Figure 2.3: (a) The colored squares represent the eyeblinks of the four viewer monkeys on each trial (blue= monkey QT, red=monkey RI, green=monkey RU, and yellow = monkey ZI). The dotted vertical lines represent the beginning and the end of the video. Frames from this video show the behavior of the stimulus monkey immediately prior to the cluster of blinks (marked by gray bars). The blinking probability of three of the four viewers increased in response during two time segments in this video. In the first segment, the stimulus monkey stared insistently at the viewer, a behavior considered as an assertion of dominance or covert threat. The second cluster of blinks occurred when the same animal began displaying a lip smacking (appeasing) expression with direct gaze. At this time in the video the stimulus monkey also blinked. Note that monkey QT systematically blinked after the presentation of the videos (clustering of blue marks at the termination of the video).

pression; two of the four monkeys blinked more often while looking at the direct gaze of a stimulus monkey with an appeasing facial expression (permutation test, $p < 0.05$, see methods; Figure 2.5).

Finally the temporal relationship was computed between the blinking of the stimulus monkey and the blinking of the viewer monkeys. This phenomenon, called eyeblink entrainment, requires the viewer to blink concurrently with its social partner (within 500 ms). Two of the four monkeys (QT and RI) entrained their eyeblinks to

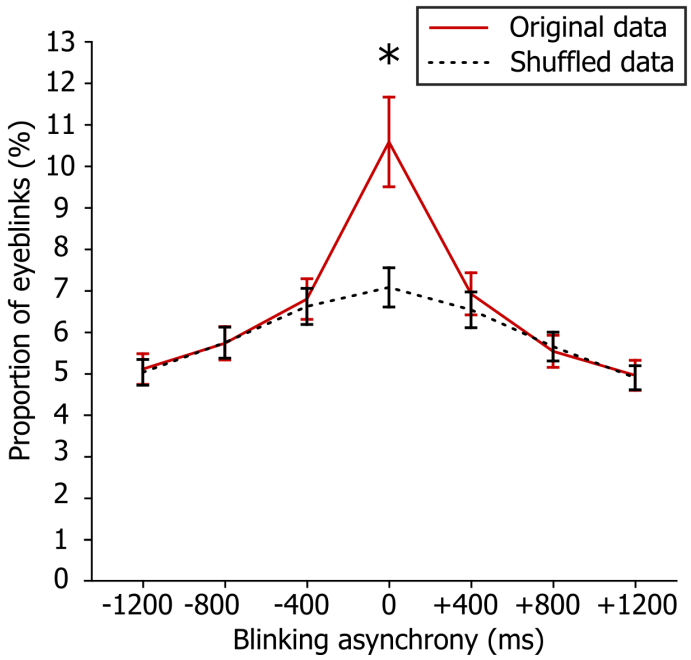


Figure 2.4: Probability of eyeblink clustering across all four subjects, based on the viewing of 1,615 videos. The solid red line represents the proportion of eyeblinks that occurred within windows of time of 400 ms during repeated presentations of the same video. The dashed line represents the same proportion for shuffled eyeblink data (see methods). The difference between the two curves reflects the degree of eyeblink synchronization (* two-way ANOVA on 7 bins: $p = 0.024$ (F=5.06) for shuffling, $p = 9.37 * 10^{-18}$ (F=15.5) for asynchrony and $p = 0.0015$ (F=3.60) for interaction). The post hoc two-tailed t-test at the central bin showed a significant difference (p=0.0030) between the actual and shuffled data. No significant difference was found at any of the other time bins. Error bars represent SEM.

the stimulus monkey's blinks (permutation test, $p < 0.05$; Figure 2.6a and 2.6b). The other two subjects (RU and ZI) did not exhibit eyeblinks entrainment (the blinking rate of these monkeys did not exceed levels expected by chance, where chance values are based on a 95% confidence interval based on shuffled data; Figure 2.6c and 2.6d). Monkeys RU and ZI were also less likely to look at the eyes of the stimulus monkey (Wilcoxon signed rank test: QT vs. RI $p = 1.68 \times 10^{-19}$; QT vs. RU $p = 9.05 \times 10^{-23}$;

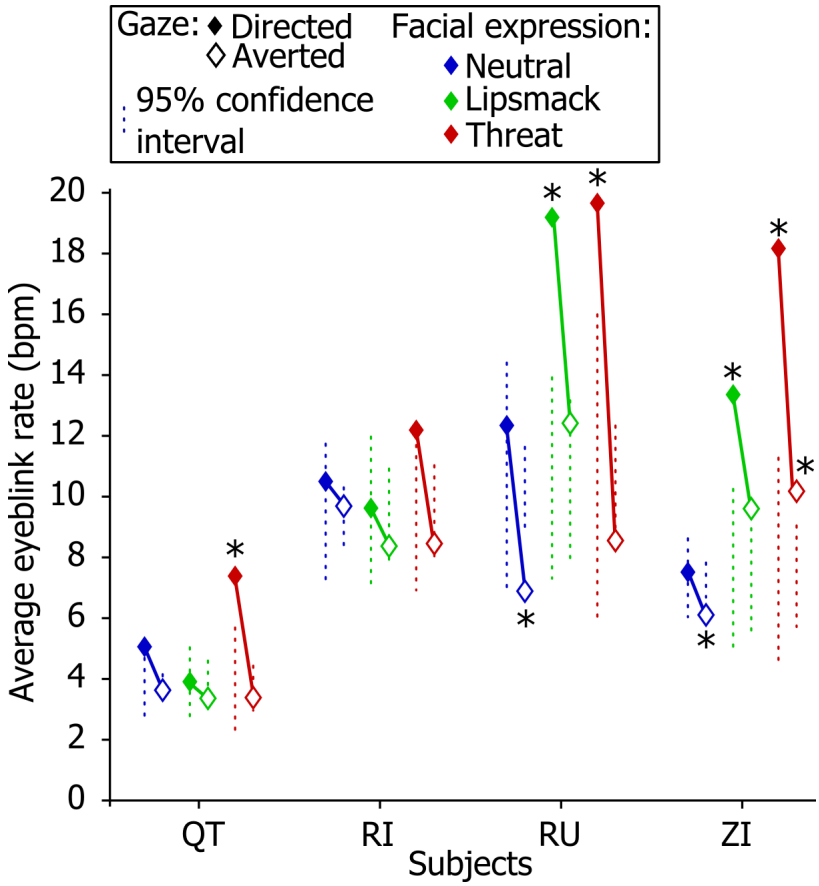


Figure 2.5: Viewer monkeys blink more frequently for facial expressions with direct gaze. The average eyeblink rate of each viewer monkey was calculated during epochs when the stimulus monkey displayed a facial expression (neutral, lip smack, or threat) and directed or averted its gaze toward or away from the viewer. Each vertical dotted line represents the 95% confidence interval calculated from shuffled data (see methods). The diamonds indicate the mean value of the eyeblink rate. Neutral faces (in blue) with either directed (filled diamonds) or averted gaze (open diamonds) did not elevate the blinking rate above the value expected by chance. Threatening (antagonistic) and lip smacking (affiliative) expressions however, significantly elevated the blinking rate of the viewer (permutation test, $p < 0.05$) with the exception of monkey RI who did not respond to any facial expressions with additional blinking. Asterisks refer to values that are outside the 95% confidence interval.

QT vs. ZI $p = 2.08 \times 10^{-21}$; RI vs. RU $p = 2.09 \times 10^{-4}$; RI vs. ZI $p = 0.029$; RU vs. ZI $p = 0.052$; Figure 2.6e). In contrast, monkeys QT and RI, who showed eye-

blink entrainment, reciprocated eye contact, by looking longer at the directed rather than at the averted eyes of the stimulus monkeys (Wilcoxon signed rank test: QT $p = 0.001$, RI $p = 3.98 \times 10^{-5}$, RU $p = 0.76$, and ZI $p = 0.29$; Figure 2.6e).

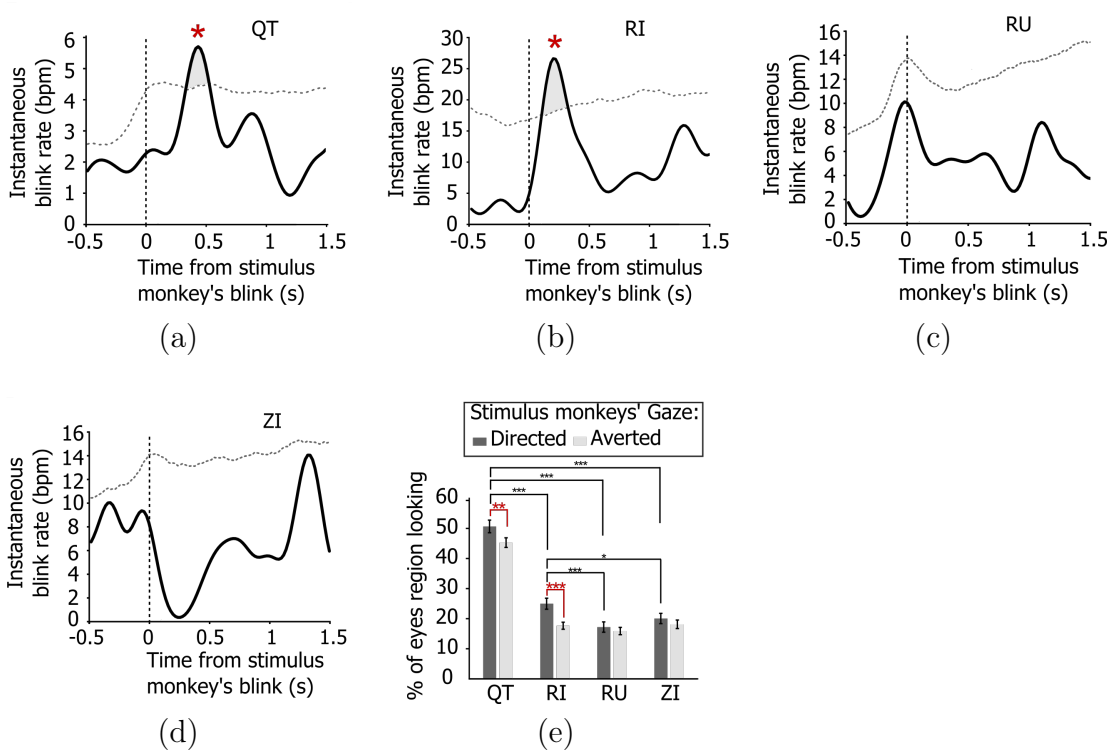


Figure 2.6: Eyeblink entrainment induced by videos with social content. Each line plot (a, b, c, and d) shows the average instantaneous blink rate of the four viewer monkeys aligned to the eyeblinks of the stimulus monkey. The vertical dotted line (time zero) represents the eyeblinks of the stimulus monkey. The horizontal dotted curve represents the boundary of the 95% confidence interval for blink rate calculated from shuffled data (see methods). Asterisk indicate significant ($p < 0.05$) increases in blinking rate. (d) Monkeys QT and RI looked longer at the eye regions of the stimulus monkeys with directed gaze (eye contact) than with averted gaze. RU and ZI, however did not look longer at eyes with direct gaze and looked less at the eyes overall compared to QT and RI. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, Wilcoxon signed rank test). Error bars represent SEM. Blink entrainment occurred within 500 ms after the eyeblinks of the stimulus monkey.

2.2.2 Discussion

The blinking behavior of four monkeys was examined while they viewed videos of conspecifics displaying facial expressions with directed or averted gaze. It was discovered that all four monkeys blinked less during the presentation of videos than during baseline periods. Even though monkeys blinked less during videos, their blinks became more temporally aligned to specific events in the video such as the production of facial expressions and the blinking of the stimulus monkeys.

During eyeblinks visual input is interrupted for about 200 ms.[90] A voluntary suppression of blinking might thus indicate a need to increase the gathering of visual information.[77,89] Indeed, we observed a reduction of blinking during the videos relative to the baseline. This reduction in blinking was strongest when monkeys viewed videos of multiple monkeys in natural social settings. It is likely that the more visually rich videos better captured the viewer's attention. This interpretation is congruent with findings that show an inverse relationship between blinking rate and attention in humans.[77,91]

The observed increases in eyeblink rates in response to facial expressions might reflect a process of overriding attentional needs by ongoing socio-emotional processes. Judicious social decisions require monkeys to process quickly and efficiently large amounts of visual information. Closing the eyes, even for the duration of an eyeblink, has been shown to help coping with increased cognitive load.[92,93] This might explain the significant increase in blinking rate that occurred in response to the segments of

the video in which the stimulus monkeys displayed threatening or appeasing facial expressions directed at the viewer. It is possible that blinking in these situations reduces not only processing demands, but also the subjective, emotional impact of these potent social signals. The observation that different viewer monkeys tend to blink in response to the same video segment supports the idea that blinks might punctuate the flow of information during socially meaningful interactions.[94]

Three of the four monkeys increased their eyeblink rate in response to threatening or appeasing facial expressions with direct gaze. The blink rate of the fourth monkeys was just marginally significant (at 96.4%, where 97.5% is the upper limit of the two-tailed test). Averted gaze, did not cause a similar increase in blink rate in any of the four monkeys, suggesting that direct gaze has a stronger effect on social behavior than averted gaze[79] enabling either social avoidance or approach.[95] This is also consistent with the finding that direct gaze activates, in the amygdala, a set of neurons singularly tuned to eye contact[87] and that patients with amygdala damage rarely make eye contact during face-to-face social interactions.[96] The biological basis and the potential functions of these changes in blinking behavior during social contact remain to be better understood.

The eyeblink entrainment reported here is highly similar to the eyeblink entrainment reported in humans.[83,84] In humans, eyeblink entrainment is not a mere imitation of others' blinks[83] rather, it is considered a marker of ongoing, fully-engaged social interactions. It follows that at least two of the subject monkeys were socially

engaged with the perceived social partner in the videos. Indeed, the two monkeys that showed eyeblink entrainment also looked longer at the eyes of the stimulus monkeys, more often reciprocating their direct gaze. Looking insistently at the eyes and returning eye contact are indicative of dominant social status in macaque societies.[97] The failure of the viewer monkeys to reciprocate the blinks of their social partner might represent an active form of avoiding social engagement with a dominant individual. It might be possible therefore to use individual variations in eyeblink entrainment as a measure of the viewer's subjective assessment of his or her status relative to the social partner. It would be interesting to determine whether the timing and rate of eyeblinks during social interactions could be added to the list of behaviors currently used for status and personality assessments in monkeys.[98–101]

In summary, macaques monkeys, like humans, blink less while they visually attend to eventful videos.[74,77,91] While the global rate of blinking was reduced, the timing of the blink appeared to mark events in the video that carried significant social weight.[78,94] Monkeys also showed blink entrainment as an elemental form of social engagement.[83,84] These findings support the view that blinking behavior of monkeys, particularly during social interactions, can be used as a measure of the ongoing socio-cognitive states.

2.2.3 Methods

2.2.3.1 Subjects and Stimuli

Behavioral data were collected from four adult male rhesus macaques (Macaca mulatta): QT, RI, RU, and ZI. At the time of the study the ages of all animals varied between 6 and 12 years. Monkeys were housed in double-size cages in the same room with visual access to all other monkeys in the colony. All experiments were performed in compliance with the guidelines of the National Institute of Health for the use of primates in research and were approved by the Institutional Animal Care and Use Committee at the University of Arizona. For accurate eye tracking each monkey was fitted with a head-fixation ring, which attached at three points to titanium pins embedded in an implant. The implant was attached to the skull by a surgical procedure under isoflurane anesthesia. Subject monkeys were seated in custom built primate chairs with their eyes positioned 57 cm from an LCD monitor spanning 37x28 degrees of visual angle (dva). Videos subtended 26x18 dva, contained 299 frames shown at 30 frames per second and contained no cuts. Neurobehavioral Systems Presentation software (Albany, CA) was used for the display of the videos. Prior to each experimental session monkeys were calibrated by fixating on a nine-point calibration grid. Errors were within ± 1 dva.

The data were collected across five years of similar experimental protocols all involving passive viewing of social videos. The duration of each video was 10 seconds;

during this time an unfamiliar monkey (stimulus monkey) displayed at least one or more threatening, neutral, or appeasing facial expressions accompanied by the corresponding postural changes. Each video contained multiple repeats of the same facial expressions with gaze either directed at or averted from the viewer. A trial (the presentation of a video) was preceded by the display of a central visual cue that remained on the monitor for 1.150 ± 250 ms. After presenting the cue, there was a 600 ± 200 ms period when the monitor was blank. The animals were not required to maintain their gaze within the boundary of the video to be rewarded. The inter-trial interval was 9.7 ± 3.3 seconds. Under our experimental conditions, it was crucial to exclude from the baseline measurement any task-related burst of eyeblink (e.g. after the presentation of the visual cue that preceded the videos or after the end of the video viewing). We thus calculated the baseline during the long inter-trial intervals, (between 7.5 seconds post-video viewing to 2.5 seconds before the next video viewing) and the intervals between the presentations of blocks of videos that spanned several minutes when the monitor was blank. The video content was ethogrammed frame-by-frame to record: direction of gaze (averted or directed at the viewer), blinks, and facial expressions (neutral, lip smacking or threatening[102]). The ethogram also recorded the number of monkeys in the frame and the background (indoors or outdoors). The videos were recorded in different environments marked in Figure 2.2 as "indoors" and "outdoors" for semi-free ranging animals and wild macaques. The frames in which the stimulus monkey's eyes were more than half-

covered by the eyelids were scored as the part of an eyeblink. Fixations on the eye region were classified based on regions-of-interest boundaries manually outlined using custom-written scripts in Matlab R2016A (Mathworks).

2.2.3.2 Eyeblink and Eye Position Measurement

Eye position and pupil diameter were recorded using an infrared camera with a sampling frequency of 240 Hz (ISCAN Inc., Woburn, MA) and collected as an analog signal using a CED Power 1401 data acquisition system and Spike 2 software (Cambridge Electronic Devices, UK). Eyeblinks were detected by a custom written script that analyzed pupil diameter, illustrated in Figure 2.7. Short, reversible losses of pupil data were identified as eyeblinks (when the eyelids were closed and the pupil was no longer exposed to the infrared beam, and the eye tracking system defaulted to maximum voltage). The pupil diameter data were smoothed with a 15 ms sliding window and a second derivate of the pupil diameter signal was taken to find the deflections (valleys) that corresponded to potential eyeblinks. The baseline signal level prior to each valley was determined to be the lower of the two highest points from either side of the valley within a 200-400 ms window. The depth of the valley was defined as the difference between the baseline and the minimum value of the valley. Two straight lines were fitted to the signal between the one third and the two third point depths on each side of the valley. The duration of the blink was defined as the length of the section between the intersections of the fitted lines with

the baseline. Valleys in the signal were considered to be eyeblinks if their duration was in the range of $20 - 800$ ms. The minimum duration between the beginnings of two consecutive eyeblinks was 200 ms. This method was manually verified using a video recording of the viewer monkey's face, with 94% match between the automated system and manual identification on a random video sample.

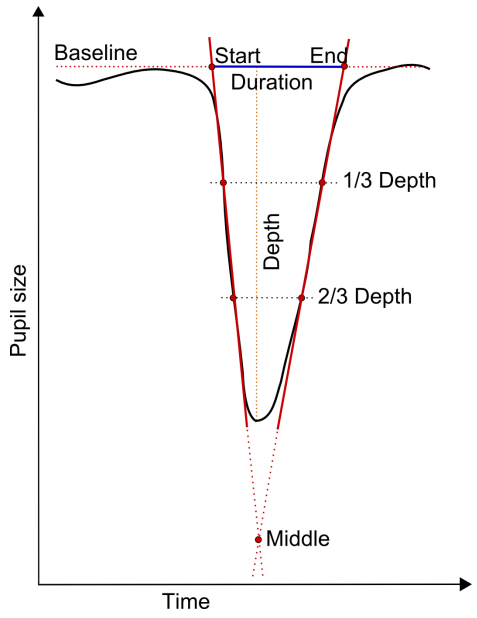


Figure 2.7: The baseline level of each eyeblink was determined as the lower of the two highest points from either sides of the valley within a $200-400$ ms window depending on the subject. The depth of the valley was defined as the difference between the baseline and the minimum value of the valley. Straight lines were fitted to the 1/3 and 2/3 depth of the valley and two fitted lines were drawn through those points. The duration of the eyeblink was defined as the length of the section determined by the intersections of the fitted lines with the baseline. The time of the blink was defined as the time of the deepest point in the valley.

2.2.3.3 Data Analysis

All data analysis and statistics were performed using custom-written scripts in Matlab R2016A (Mathworks). To account for individual differences in blink rates, we calculated the mean blink rate during the movie and during baseline periods. Baseline periods began 7.5 seconds after the termination of each video and ended 2.5 seconds prior to the presentation of the next video. The intervals between blocks of videos, when the monitor was blank and typically spanned several minutes, were also included the calculation of baseline blinking rates.

To assess the temporal clustering of blinks among viewers (Figure 2.4), we adopted a method previously used by Nakano and colleagues (2009).[94] We calculated the shortest time interval between a blink in a given presentation (reference) and all of the other blinks in each different presentations of the same video (test). These time differences were binned into 400 ms bins. The same procedure was then applied to surrogate data obtained by shuffling blink times. We obtained the shuffled blink data by shifting all the blinks within a trial by a random time (with circular boundary conditions). This form of shuffling preserves the natural blink rate of the monkey but disrupts the relationship between the blinks and the content of the videos.

To establish a correlation between the viewer's blink rate and the stimulus monkey's facial expression, we marked the frames that contained neutral, appeasing (lip smacking) and threatening (open-mouth threat) facial expressions. We also marked for each frame the gaze direction of the monkey shown in the video and whether

the viewer monkey was looking at the video. We then compared the average blink rate during each expression and gaze direction combination to the blink rate during re-sampled, time-matched video segments. We only included in the analysis video segments when the viewer monkey was gazing at the eyes of the viewer monkey. We calculated 2000 shuffled time-matched segments and determined whether the blink rate during each facial expression fell outside the 95% confidence interval (two-tailed test).

Eyeblink entrainment was quantified in two steps. First we calculated the average instantaneous blink rate of the viewer monkey relative to the blinks of the stimulus monkey. The instantaneous blink rate, was calculated based on a formula used previously by Shultz et al. 2011:[91]

$$b(t) = \frac{T}{\sigma\sqrt{2\pi}} \sum_i e^{-(t-t_i)^2/(2\sigma^2)} \tag{2.1}$$

where $b(t)$ is the time-dependent instantaneous blink rate function and t_i are the blink times. The standard deviation of the Gaussian kernel was chosen as a fixed value of $\sigma = 100$ ms.[103] We used $T = 60$ s to express the results in blink-per-minute (bpm) units.

Second, we determined whether the observed eyeblink entrainment was significantly different than expected by chance, we generated a reference dataset by replacing the blinks of the stimulus monkey with the same number of uniformly distributed randomly generated blinks. This randomization process was repeated 3000 times

to yield 3000 different peri-event time histograms. The observed eyeblink entrainment was then compared to the 95% confidence interval calculated from these 3,000 surrogate peri-event time histograms (one-tailed comparison, looking for eyeblink entrainment that was significantly higher than chance).

We included in the analysis only the trials in which the viewer looked at the video for at least 200 ms (the duration of 1-2 fixations) before the stimulus monkey blinked. We included this criterion to be certain that the viewers were attending to the stimulus monkey and thus, noticed the stimulus monkey's blink. During the first 300 ms in the plot the viewer may or may not be looking at the eyes of the stimulus monkeys. Of the monkeys that looked frequently at the eyes (e.g., QT), the confidence interval calculated from the shuffled data appears to be low 500 ms before the stimulus monkey's eyeblink and then gradually rises to a stable value by time point 0 ms (Figure 2.6a). This is due to our 200 ms eye-looking limit (this also explained why the shuffled data/upper limit of the confidence interval is not straight).

Given that monkeys have high levels of blink suppression during the first trial, we excluded the this trial when analyzing eyeblink entrainment. Likewise, given that monkeys spend less time looking at the videos after several repeated exposures, they are less likely to see the eyeblink of the stimulus monkey. To account for this, we only included trials up to the fifth viewing. We also eliminated from the analysis 10% of trials where the viewer monkey spent the most time looking at the screen and 10% of trials where the viewer monkey spent the least time looking at the screen (often the

last trial).

2.3 Bayesian Changepoint Detection for Identification of Changes in Neural Firing Rates

While analyzing the role of blinking as a social indicator, we noticed a neural response to blinking exhibited by some of the amygdala neurons. As seen in Figure 2.8, multiple neurons collected from the same experiment described in Section 2.2.3.1 demonstrate a change in neural firing rate during the monkey's blinks. Although many of the neurons that exhibit a change in firing rate corresponding to the time of the blinks are visually obvious in their response, having an algorithm to automatically detect these changes would not only reduce analysis time, but also potentially detect neurons with a firing rate change that is not visually perceptible. This section details the implementation of a Bayesian changepoint detection algorithm to determine which neurons exhibit a change in firing rate as a result of the blinks.

As previously stated, the neural data analyzed here comes from the same experiment described in Section 2.2.3.1. The data comes from three monkeys (ZI, QT, and GI) measured over the course of four years while participating in a social experiment where monkeys watched videos of unfamiliar monkeys in socially meaningful situa-

tions. In total 341 neurons were used in this analysis, coming from five different nuclei (accessory basil, basil, lateral, medial, and central) of the amygdala. For each neuron, its nuclei location was known. Eye tracking data was used to determine the time of the blinks using the method described in Section 2.2.3.2. Using visual inspection, 83 of the 341 neurons were determined to exhibit a change in firing rate during the time of the blink, which provided a ground truth for this work.

2.3.1 Changepoint Detection Algorithm

This analysis implements the Bayesian changepoint detection algorithm for online inference as described by Adams and MacKay.[104] In this algorithm the posterior distribution is estimated over the current run length. When the algorithm begins it assumes there has just been a changepoint, so the current run length is equal to zero. At each time point, the current observation is considered which can either increase the current run length by one, or result in a changepoint, returning the run length to zero. The conditional prior on the changepoint can be stated as:

$$P(t_t|r_{t-1}) = \begin{cases} H(r_{t-1}+1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1}+1) & \text{if } r_t = r_{t-1}+1 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

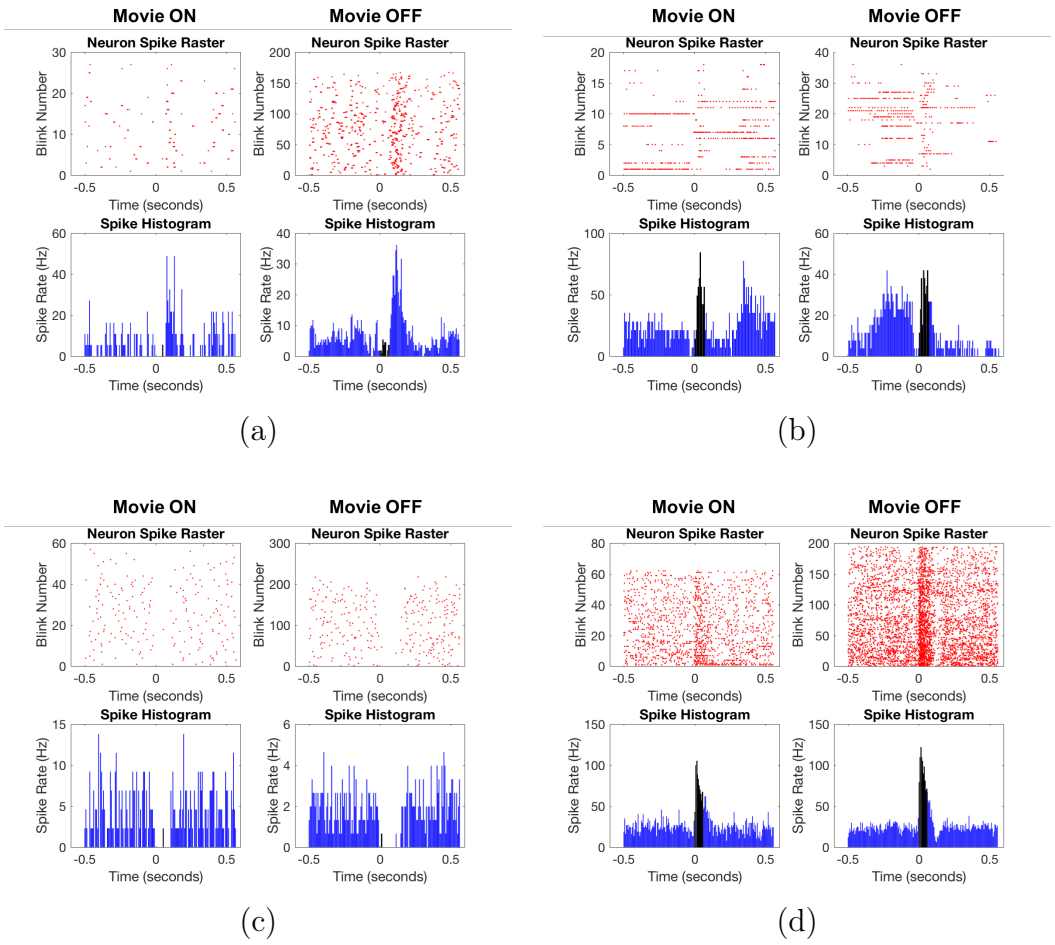


Figure 2.8: Neurons with a visually obvious change in firing rate during the duration of the blink. Plots are shown both from blinks that occurred while the monkey watched the movies, and blinks during the movie inter-trial period. (a-d) illustrate four different neurons with firing rate changes during the blinks. Average blink length is colored in black.

where the function $H(t)$ is the hazard function, given by:

$$H(\tau) = \frac{P_{gap}(g = r)}{\sum_{t=\tau}^{\infty} P_{gap}(g = t)} \tag{2.3}$$

If $P_{gap}(g)$ is geometric with timescale λ, then the process is memoryless and the

hazard function becomes $H(\tau) = 1/\lambda$. Because the predictive distribution only depends on the recent data, recursive message-passing can be used for the joint distribution over the current run length and the data based on the prior over r_t given r_{t-1} and the predictive distribution over the most recently observed datum given the data observed since the last changepoint. Lastly, a Viterbi algorithm was used to back calculate the locations of the changepoints from which it is determined if there has been a change in firing rate in the data in response to the blink.

2.3.2 Preprocessing Steps

To perform changepoint detection on each neuron's data, the spikes from each neuron were converted into a signal that assumed the data either came from a Poisson or Gaussian distribution. To form the signal with data from a Poisson distribution, the neural spikes within the time window of one second before each blink and one second after each blink were binned (bin sizes varied from 0.05 seconds, 0.01 seconds, 0.005 seconds, and 0.001 seconds). For each bin, the total number of spikes that occurred in that bin for all blinks was computed. This was done for each non-overlapping bin in the one second before the blink and one second after the blink time window. The sums of each of the bins form the signal upon which changepoint detection was performed.

To form the signal that assumed the data came from a Gaussian distribution, the same window of spikes one second before each blink and one second after was analyzed.

A sliding window (window sizes included 0.05 seconds, 0.01 seconds, 0.005 seconds, and 0.001 seconds) was passed over the spikes surrounding each blink, calculating the average number of neural spikes in each window. Windows overlapped by half of their size and the signal was normalized before analysis. Once created, changepoint detection was performed on this signal.

2.3.3 Parameter Initialization

There are a number of parameters requiring initializing in this algorithm. For both the signals based on a Poisson distribution and the signals based on a Gaussian distribution, the hazard function is a function of λ which is a parameter that can be varied to increase accuracy. λ corresponds to the probability of a changepoint occurrence and increasing λ decreases the hazard function, thus decreasing the probability of a changepoint occurrence. Additionally, both distributions had prior distributions that required parameter initialization. For the Poisson distribution signal implementation, a Gamma distribution was used for the prior with parameters α and β. Because $\mu = \alpha/\beta$, α was set equal to the mean value of the Poisson distributed signal, and β was set equal to 1. For the Gaussian distribution signal implementation, a normal Gamma distribution was used for the prior which has parameters μ, α, and β, which were set to 0, 1, and 1, respectively.

2.3.4 Parameter Tuning

To determine the ideal bin size and value for λ, changepoint detection was run on signals created with different bin sizes, while also using different values for λ. For this analysis, it is assumed that there is a change in firing rate in response to the blink when there were three or less changepoints in the time interval 0.4 seconds before to 0.4 seconds after the blink onset, and there were no other changepoints in the rest of the time interval, i.e. [-1, -0.4) and (0.4, 1]. The algorithm assumes a changepoint at the beginning of the sequence (t = -1) but this changepoint is not considered for this determination.

Using this criterion, Figure 2.9 gives the false positive ratio and true positive ratio for the different bin sizes and values of λ for both changepoint detection on the signals from the Poisson distributed data and Gaussian distributed data. If the criterion is changed to allow for zero or one changepoints in the time window (-1, 0.4] and (0.4, 1] then the true positive accuracy for some signals with Poisson distributed data jump to values just below 70% and the true positive accuracy for some signals with Gaussian distributed data jump to values just below 60%. These plots are shown in the bottom of Figure 2.9.

Figure 2.10 shows for each bin size and value of λ how many neurons out of the original 341 neurons had a neural firing rate change in response to the blink detected by changepoint detection. The top subplots show the neurons detected from the pool of all 341 neurons, and the bottom subplots shows the neurons detected from the pool

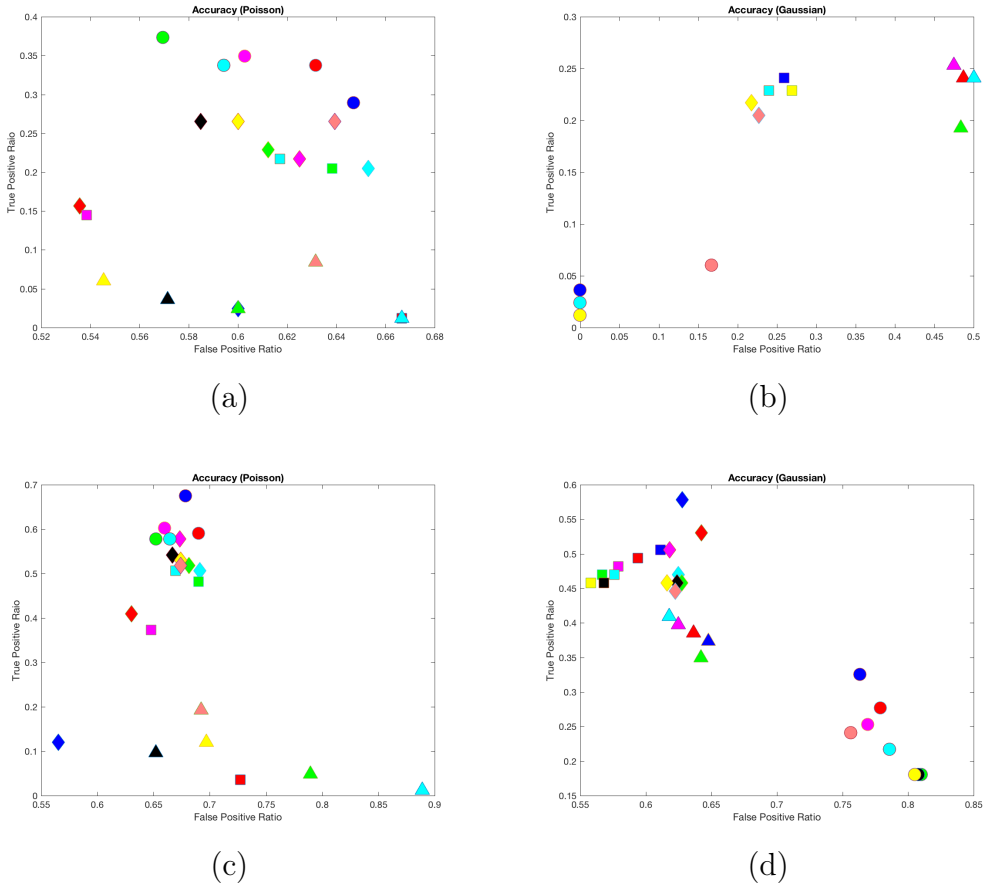


Figure 2.9: False positive and true positive rates for signals with (a) Poisson distributed data and (b) Gaussian distributed data, created using different bin sizes (0.001 seconds, 0.005 seconds, 0.01 seconds, 0.05 seconds) and lambda values (20, 30, 50, 70, 80, 90, 100, 120). Each color corresponds to a value of lambda (20 = blue, 30 = red, 50 = magenta, 70 = cyan, 80 = green, 90 = black, 100 = yellow, 120 = light red), and each marker shape corresponds to a bin size (0.001 = triangle, 0.005 = square, 0.01 = diamond, 0.05 = circle). (c) and (d) show the results for the signals with (c) Poisson distributed data and (d) Gaussian distributed data when the firing rate change criterion instead allowed for three or less changepoints in the time interval (-0.4, 0.4) and one or less changepoints elsewhere.

of the 83 neurons determined to respond to the blinks by visual inspection. From these plots, a bin size of 0.05 seconds and a λ value of 50.0 was chosen for analyzing the signals with Poisson distributed data, and a window size of 0.001 seconds and

λ value of 50.0 was chosen for analyzing the signals with Gaussian distributed data. These values were chosen in an attempt to minimize false positives and maximize true positives. Subsequent analysis used these parameter values unless otherwise stated.

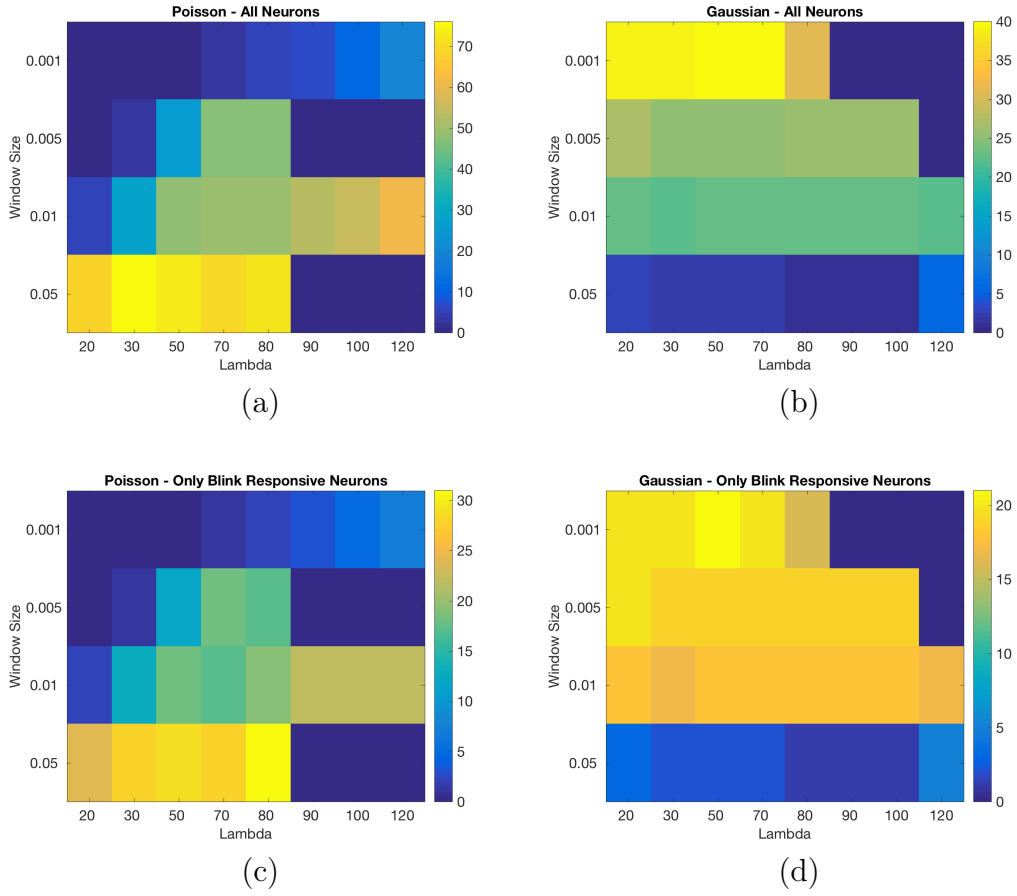
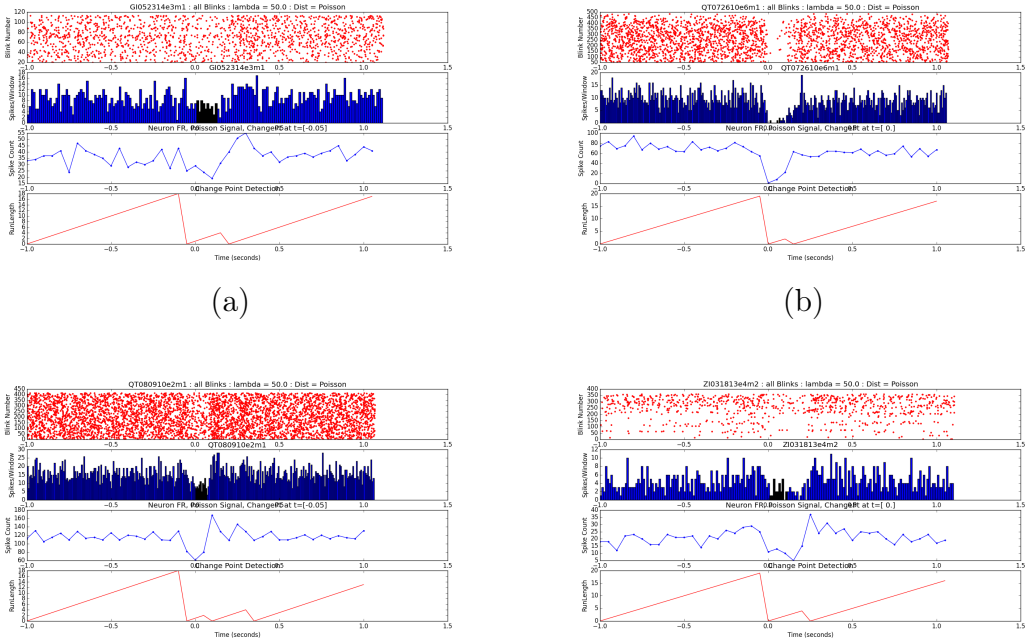


Figure 2.10: Indicates the number of neurons whose blink-responsiveness is correctly detected using changepoint detection. The corresponding lambda value and bin size are shown. Plots are shown for the signals with (a) Poisson and (b) Gaussian distributed data, as well as just for the signals with (c) Poisson and (d) Gaussian distributed data created from the neurons determined to respond to blinks by visual inspection.

2.3.5 Changepoint Detection Accuracy

Results were determined by using a bin size of 0.05 seconds and a value of 50.0 for λ to make the signals with Poisson distributed data, and a window size of 0.001 seconds and value of 50.0 for λ to make the signals with Gaussian distributed data for all neurons. For the signals with Poisson distributed data, changepoint detection yielded 83 neurons that exhibited a firing rate change in response to the blink, 33 of which were included in the list of neurons determined to respond to blinks by visual inspection. This resulted in a false positive ratio of 60.3% and a true positive ratio of 34.9%. Some examples of these neurons can be seen in Figure 2.11.



(a) (b)

(c) (d)

Figure 2.11: Examples of changepoint detection to detect changes in firing rate in response to eyeblinks using a signal with Poisson distributed data generated from the neural spiking data.

For the signals with Gaussian distributed data, changepoint detected yielded 40 neurons that exhibited a change in firing rate in response to the blinks, 21 of which of which were included in the list of neurons determined to respond to blinks by visual inspection. These signals results in a false positive ratio of 47.5% and a true positive ratio of 25.3%. Some examples of these neurons can be seen in Figure 2.12.

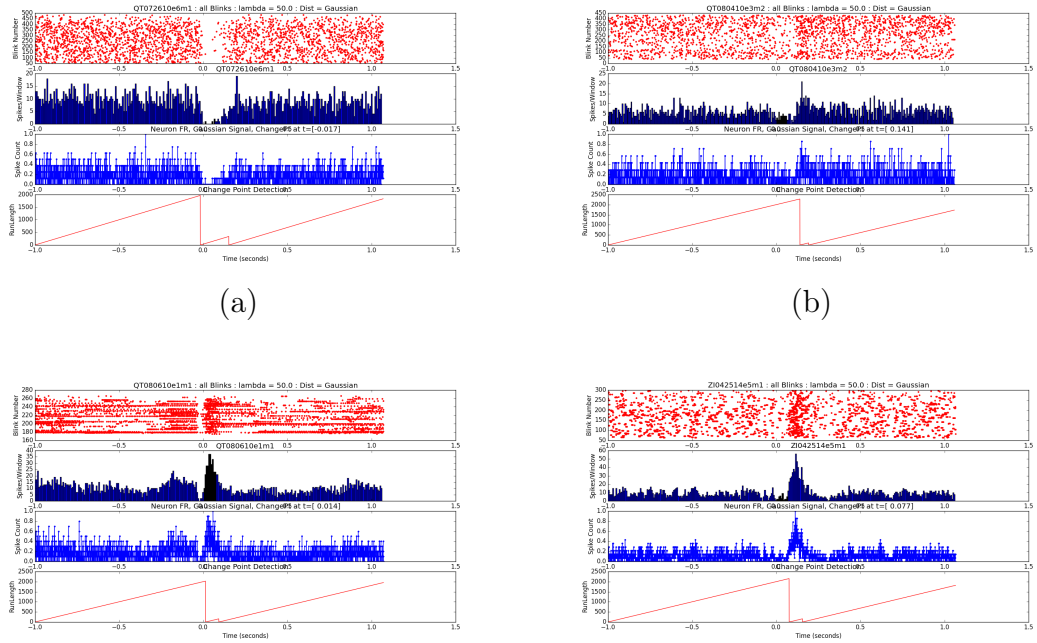


Figure 2.12: Examples of changepoint detection to detect changes in firing rate in response to eyeblinks using a signal with Gaussian distributed data generated from the neural spiking data.

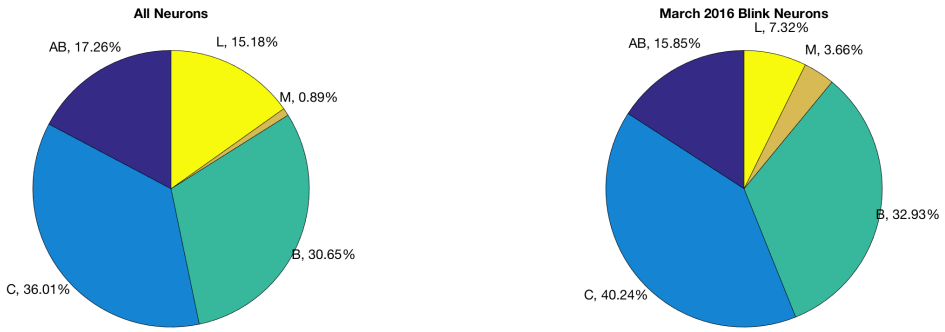
23 neurons indicated a change in firing rate in response to the blink detected by changepoint detection for both the signals with Poisson distributed data and Gaussian distributed data. 12 of these 23 neurons were were determined to respond to the blinks by visual inspection. Upon visual inspection, the signals with Gaussian

distributed data appeared to provide more accurate time detection of the change in firing rates, although this makes sense since the signals with Gaussian distributed data were formed from bins of 0.001 seconds and the signals with Poisson distributed data were formed from bins of 0.05 seconds.

2.3.6 Blink-Responsive Neurons' Nuclei Distribution

Additionally it was considered in which nuclei of the amygdala the neurons with responses to blinking were located. Figure 2.13 shows the distribution of nuclei for all 341 neurons unitized in this analysis as well as the distribution of nuclei for the 83 neurons determined to respond to blinks by visual inspection. Figure 2.14 shows the distribution of nuclei for the neurons determined by changepoint detection to respond to the blinks. This figure shows the distribution of neurons for which the signals created with Poisson and Gaussian data resulted in successful changepoint detection to detect a response to the blinks, and the distribution of nuclei for the subset of neurons within those larger sets also from set of blink-responsive neurons determined by visual inspection. The figure also shows the nuclei distribution for neurons for which changepoint detection detected a response to blinks using both the signal created with Poisson distributed data and the signal created with Gaussian distributed data and not just one or the other.

Overall the distributions of nuclei are similar across all cases. The neuron subgroup determined to be blink-responsive by visual inspection contained more neurons



(a) (b)

Figure 2.13: (a) Distribution of amygdala nuclei for the 341 neurons analyzed. (b) Distribution of amygdala nuclei for the 83 neurons determined to respond to blinks by visual inspection.

proportionally (as compared to the larger 341 neuron population) from the medial (M), basal (B), and central (C) nucleus, as compared to the accessory basil (AB) and lateral (L) nucleus. The neurons for which changepoint detected yielded the correct result using a signal with Poisson distributed data, were distributed at a higher percentage from the basil (B) and medial (M) nuclei, whereas the neurons for which changepoint detection yielded the correct result with a signal from Gaussian distributed data came proportionally more from the lateral (L) and basal (B) nuclei. The neurons that exhibited successful blink changepoint detection for both signal did not come from the medial (M) nucleus, but instead their majority was located in the basil nucleus.

Nonetheless these differences were not great between the different cases. Additionally, the method described does not perform at a high enough accuracy to detect the

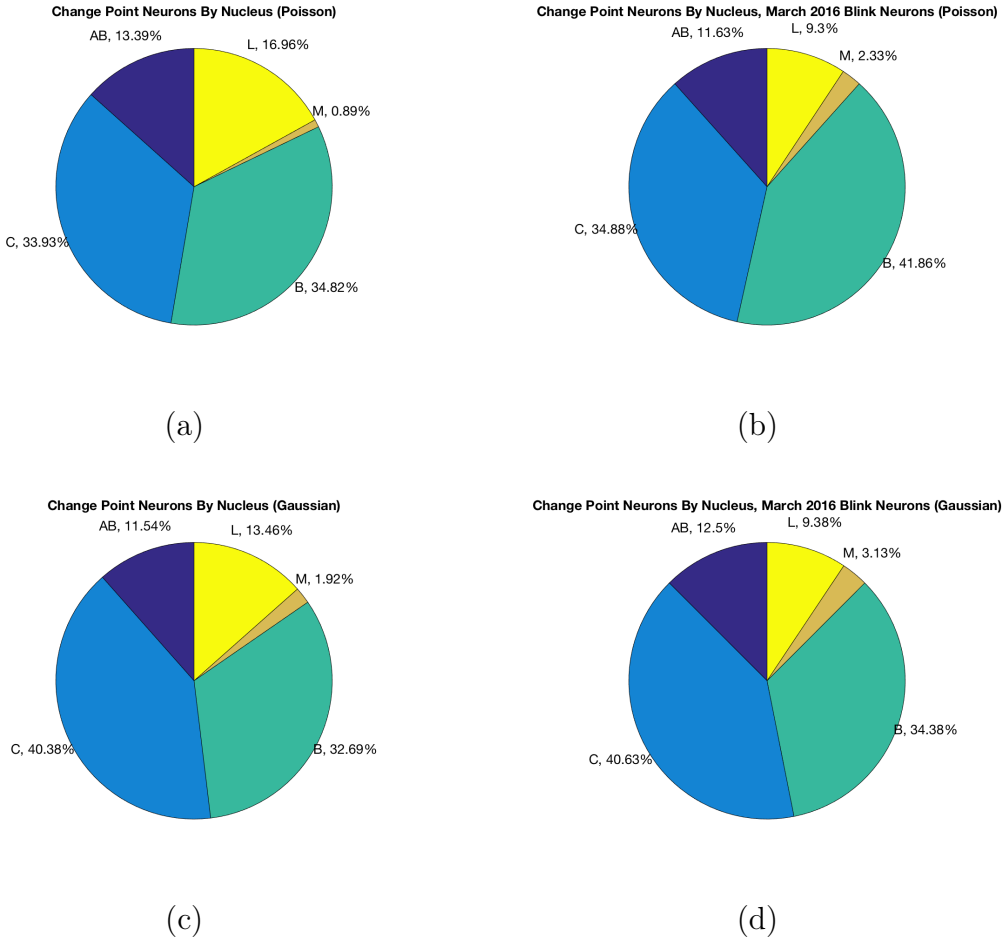
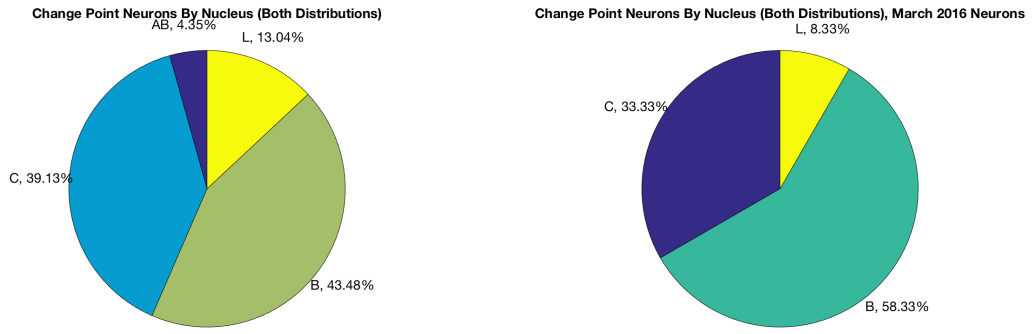


Figure 2.14: (a) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Poisson distributed signal. (b) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Poisson distributed and subgroup determined to be blink-responsive by visual inspection. (c) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Gaussian distributed signal. (d) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Gaussian distributed and in the subgroup determined to be blink-responsive by visual inspection. (d) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Poisson and Gaussian distributed signal.

majority of neurons previously determined to have a change in firing rate in response to the blink.



(e) (f)

Figure 2.15: (e) Distribution of amygdala nuclei for the neurons determined by changepoint detection to respond to the blink, using a Poisson and Gaussian distributed signal and in the and subgroup determined to be blink-responsive by visual inspection.

2.4 Point Process Modeling to Predict Neuron Firing

When analyzing neural data, often scientists are most concerned with the specific time of neuron spikes, and much of the biological details of neurons can be abstracted away. When this is the case, neuron spike trains can be modeled as point processes, producing a stochastic set of localized events in time or space. Using point process modeling the likelihood of each neuron's firing can be described based on an underlying model that incorporates measurable parameters.[105,106] When defining a point process, the signal may be conceptualized as the time of spikes, the waiting time between spikes, the increments of counts in a process, or discrete binary indicators.

For this work, the point process data is considered to be discrete binary indicators as seen in Figure 2.16.

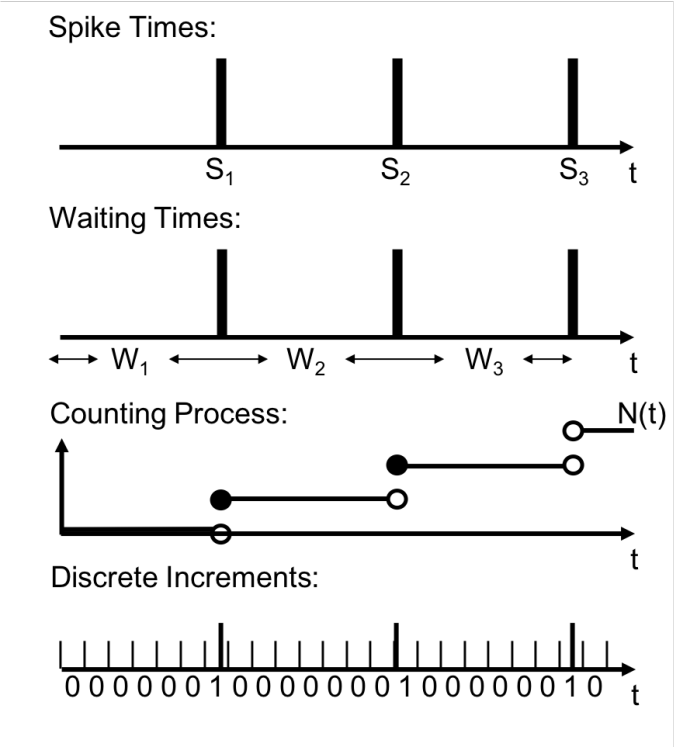


Figure 2.16: Point processes may be defined in terms of the time of spikes, the waiting time between spikes, the increments of counts in a process, or discrete binary indicators.

For this work, the data is considered to be discrete binary indicators where, each time window ΔT is reduced such that it will only contain at most one spike per window. Thus if an underlying Poisson distribution is assumed, the probability of a given spike train can be defined as:

$$p(SpikeTrain) = \prod_{u=1}^{T} p(dN_u \mid H_u) \tag{2.4}$$

$$= \prod_{t \in Spikes} \lambda(t \mid H_t)\Delta t * exp(-\lambda(t \mid H_t)\Delta t) \prod exp(-\lambda(n \mid H_n)\Delta t) \tag{2.5}$$

$$= \prod_{t \in Spikes} (\lambda(t \mid H_t)\delta t) * exp(-\sum_{u=1}^{T} \lambda(u \mid H_u)\Delta t) \tag{2.6}$$

$$= exp(\sum_{u=1}^{T} log(\lambda(u \mid H_u)\delta t) dN_u - \lambda(u \mid H_u)\Delta t) \tag{2.7}$$

The conditional intensity function of a neuron, $\lambda(t \mid H_t)$ is defined as:

$$\lambda(t \mid H_t) = \lim_{\Delta t \to 0} \frac{P(s(t, t+\Delta t)|H_t)}{\Delta t} \tag{2.8}$$

where H is the history of the spiking process up to time t, and $s(t, t+\Delta t)$ gives the spikes in the time window $(t, t+\Delta t)$.

Because the Poisson distribution belongs to the exponential family, the likelihood function can then be written in terms of the generalized linear model. By writing the likelihood in this form, one can realize the canonical link function $C(\theta)$ which is a linear function of parameters. The exponential family of distributions is defined as

$$L(\theta) = (f * y \mid \theta) \tag{2.9}$$

$$= \prod_{k=1}^{K} exp\{T(y_k)C(\theta) + H(y_k) + D(\theta)\} \tag{2.10}$$

with a canonical link function of

$$C(\theta) = \theta_0 + \sum_{j=1}^{J} \theta_j g(x_j) \tag{2.11}$$

The link function is a linear function of parameters. For Poisson data the likelihood is,

$$L(\theta) = \prod_{k=1}^{K} \frac{\lambda_k^{y_k} exp\{-\lambda_k\}}{\lambda_k!} \tag{2.12}$$

$$= \prod_{k=1}^{K} exp\{y_k log(\lambda_k) - log(y_k!) - \lambda_k\} \tag{2.13}$$

and the canonical link function is:

$$C(\theta) = log(\lambda_k) = \sum_{j=1}^{J} \theta_j x_{kj} \tag{2.14}$$

which gives the relationship between the neuron firing and measurable parameters. This is the basis for the subsequent modeling and analysis.

2.4.1 Experimental Setup

As described in Section 2.2 and Section 2.3, the data utilized to build the described point process models came from an experiment where monkeys watched movies of unfamiliar monkeys eliciting socially meaningful facial expressions. Specifically the work described in this section focused only on the data collected from one monkey, QT. This monkey participated in 16 individual sessions viewing 192 different movies over the course of all sessions. During each session, 16 - 24 different movies were watched one to ten times. On average 22 movies were watched during each session, and each movie was watched an average of five times. Movies were divided into four categories based upon the title of the movie, including "threatening", "neutral", "submissive / lip-smack", and "other". During these sessions, 133 different single unit neurons were recorded and 2 - 13 neurons were recorded during each session. Figure 2.17 shows examples of screen shots from the "threatening", "neutral", and "submissive" movies.

2.4.2 Average Firing Rate Analysis

To assess initial viability of the hypothesis that the type of movie viewed affected the firing rate of neurons in the amygdala, the average firing rate was calculated for each neuron for each category of movie. Figure 2.18 shows this for all neurons in the dataset, color coded by the nuclei of the amygdala where each neuron was located. Figure 2.19 illustrates the same data but deletes any neurons with a firing rate of

Figure 2.17: Screen shots captured from the different movie categories. The top row shows screen shots from movies categorized as "threatening", the middle row shows movies categorized as "neutral", and the bottom row shows movies categorized as "submissive/lip-smack".

less than 10 Hz from the dataset. From these plots, it was clear that there were neurons whose average firing rate differed given the category of movie that was being viewed, which encouraged further testing of the hypothesis that neuron firing rate was a function of the category of movie viewed by the monkey.

Eliminating the neurons with firing rates less than 10H z resulted in 32 neurons remaining neurons for analysis, although 2 of these 32 neurons were unable to be processed due to issues with their formatting and collection. Thus, the resulting 30 neurons were the neurons analyzed with this method. Reducing the number of

neurons, additionally reduced the initial complexity of this work. Additionally, the average firing rate for both movie duration and when there was no movie displayed was plotted, and is shown in Figure 2.20. Originally this data was included in the model but was later removed to decrease the size of the covariate matrix.

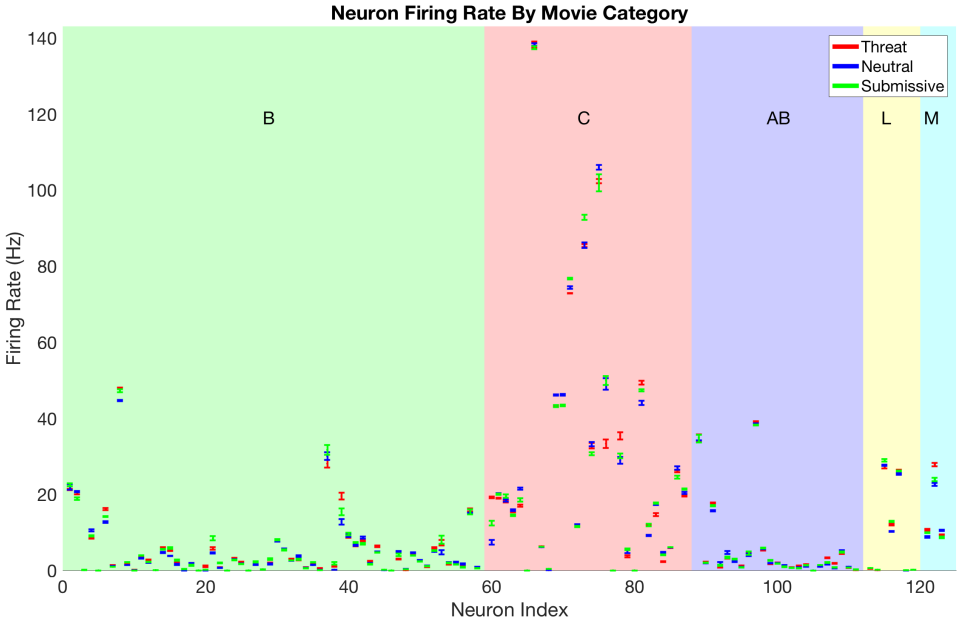


Figure 2.18: Average neuron firing rates for each movie category by neuron. Neurons are plotted by the nuclei in the amygdala in which they were located. Error bars indicate (2 * standard error).

2.4.3 Model Definition

Point process models were constructed to predict firing rate given a number of observed covariates. Using the covariates and the MATLAB function glmfit, the parameters of the canonical link function were determined for the Poisson data by

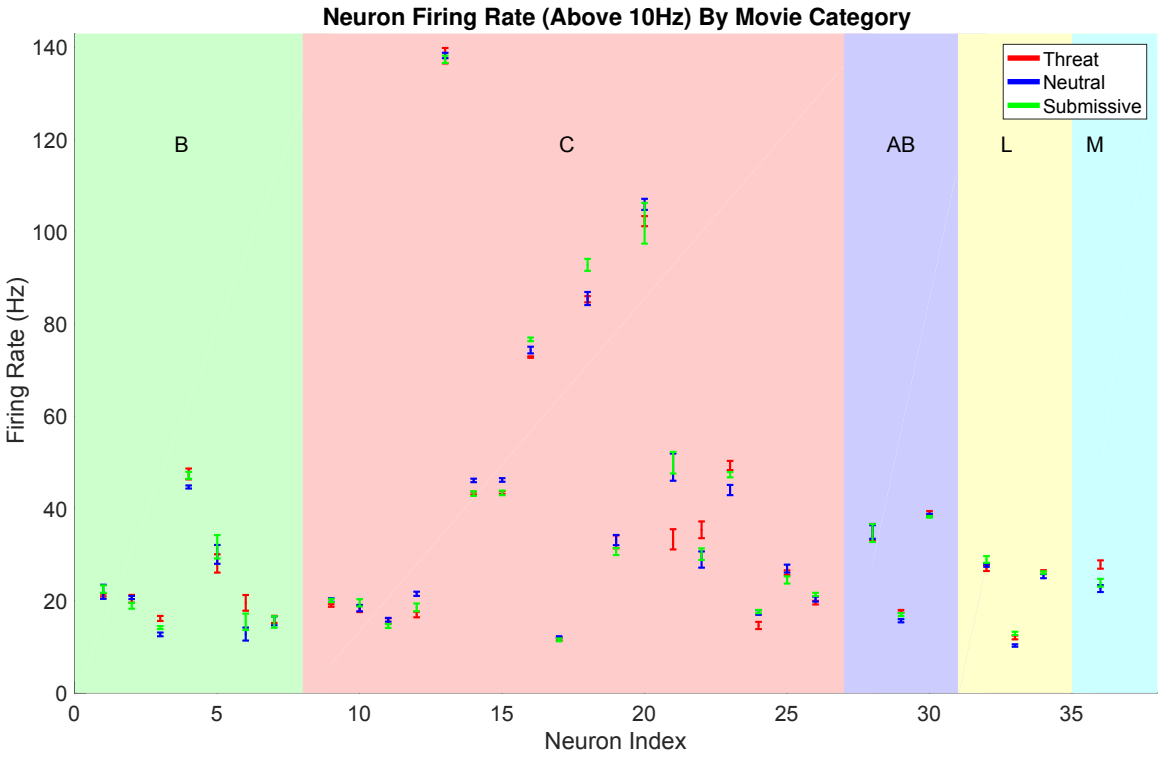


Figure 2.19: Average neuron firing rate for each movie category for only neurons with firing rates greater than 10 Hz. Neurons are plotted by the nuclei in the amygdala in which they were located. Error bars indicate (2 * standard error).

using a generalized linear model.

The following four models were devised, one for each type of movie, where $th =$ threatening movie, $ne =$ neural movie, $ls =$ lip smack movie, and $other =$ corresponds to any movie not in the previous categories.

$$log(\lambda_{th}(t \mid H(t)) = \alpha_{th} * I_{th} + \sum_{i=1}^{10} \beta_i * n_{(t-i)} + \sum_{j=1}^{27} \sigma_j * n_{((t-10)-5*(j-1) \longrightarrow (t-10)-5*j)} \tag{2.15}$$

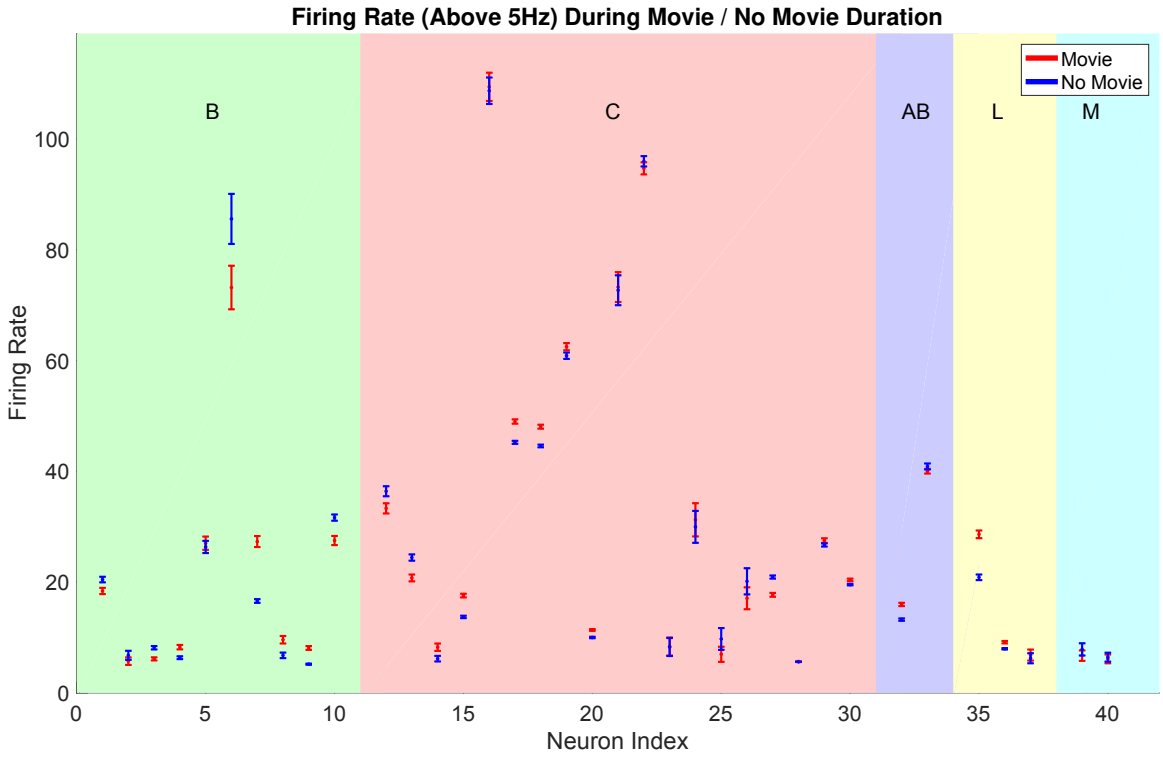


Figure 2.20: Average neuron firing rates during and between the movie durations for neurons with firing rates above 5 Hz. Error bars indicate $(2 * standard_error)$.

$$log(\lambda_{ne}(t \mid H(t)) = \alpha_{ne} * I_{ne} + \sum_{i=1}^{10} \beta_i * n_{(t-i)} + \sum_{j=1}^{27} \sigma_j * n_{((t-10)-5*(j-1) \longrightarrow (t-10)-5*j)} \tag{2.16}$$

$$log(\lambda_{ls}(t \mid H(t)) = \alpha_{ls} * I_{ls} + \sum_{i=1}^{10} \beta_i * n_{(t-i)} + \sum_{j=1}^{27} \sigma_j * n_{((t-10)-5*(j-1) \longrightarrow (t-10)-5*j)} \tag{2.17}$$

$$log(\lambda_{other}(t \mid H(t)) = \alpha_{other} * I_{other} + \sum_{i=1}^{10} \beta_i * n_{(t-i)} + \sum_{j=1}^{27} \sigma_j * n_{((t-10)-5*(j-1) \longrightarrow (t-10)-5*j)} \quad (2.18)$$

λ gave the spiking activity for the neuron. Each model had 1 model-specific parameter and 37 parameters shared across all four models. The model parameters included an α parameter that was multiplied by an indicator function to indicate the movie type, 10 β parameters that were multiplied by the short term spiking history (one millisecond bins from $t-1$ to $t-10$ milliseconds), and 27 σ parameters that were multiplied by the long term spiking history (five millisecond bins from $t-10$ to $t-150$ milliseconds). Although these models could be viewed as four distinct models, when processing in MATLAB the system was treated as one model with four α parameters that were zero when the current movie was not of that category.

2.4.4 Covariate Data

To form the covariate data matrix for each neuron, three movies from the the same category were chosen at random over the course of the same viewing session during which the given neuron was recorded. One viewing per movie was randomly chosen to be a part of the test data and a different viewing of that same movie was randomly chosen to be a part of the training data. This was done for all four movie categories. It was assumed that the neural response did not change from one viewing

of the movie to another for simplicity.

On average it took 15 minutes to make the covariate matrix for one neuron in MATLAB, which was why only data from during movies was used, and the data from the period between movie viewings was not. For future research other computational methods or resources could be employed to accelerate this process.

2.4.5 Results

2.4.5.1 Parameter Significance

Figure 2.21 shows the results of the Kolmogorov-Smirnov analysis for all 30 neurons for which models were built. For most plots, the models stayed within the 95% confidence bounds, which are shown in red. Figure 2.22 shows the learned parameters for each neuron. Each neuron had 42 parameters including 1 base firing rate parameter, 4 α parameters, 10 β parameters, and 27 σ parameters. The confidence intervals are shown, as well as a line to indicate e^0. Figure 2.23 shows the p-values of all parameters, indicating that many are significant.

Figure 2.24 shows, for each neuron, which parameters had a p-values greater than 0.05. Parameters with a p-value greater than 0.05 are indicated on the plot with a red dot. Parameters with a p-value less than 0.05 have no dot indication. Lines are drawn to indicate the α, β, and σ parameters. From this plot, it was concluded that the α parameters, which related to the type of movie were often significant to the

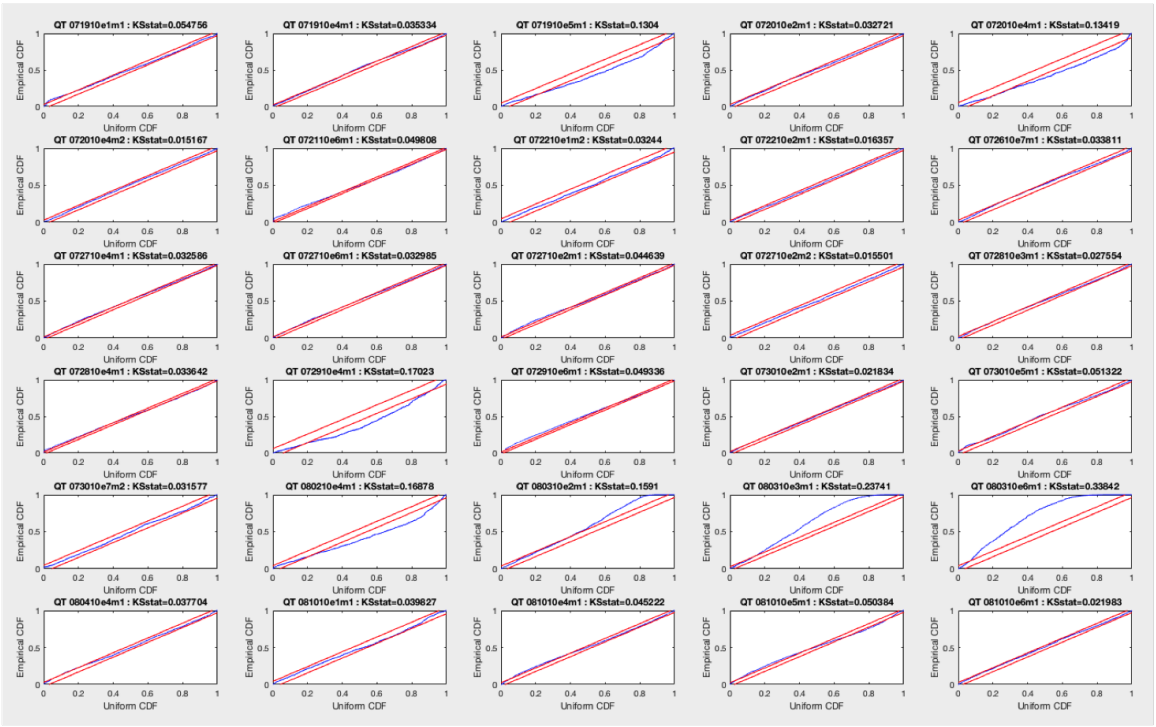


Figure 2.21: Kolmogorov-Smirnov (KS) plots for each of the 30 neurons analyzed.

model, and that long term spiking history was often more significant than short term spiking history.

2.4.5.2 Parameter Elimination Analysis

To simplify the model and determine the most significant parameters to predict neuron firing rate, the data was processed through a parameter elimination experiment. For this experiment the model begins with 30 parameters. Based on the previous analysis of parameter significance only the α parameters and any other parameter that was significant for more than ten neurons was considered.

In the first iteration of this experiment, 29 models were built with 29 different

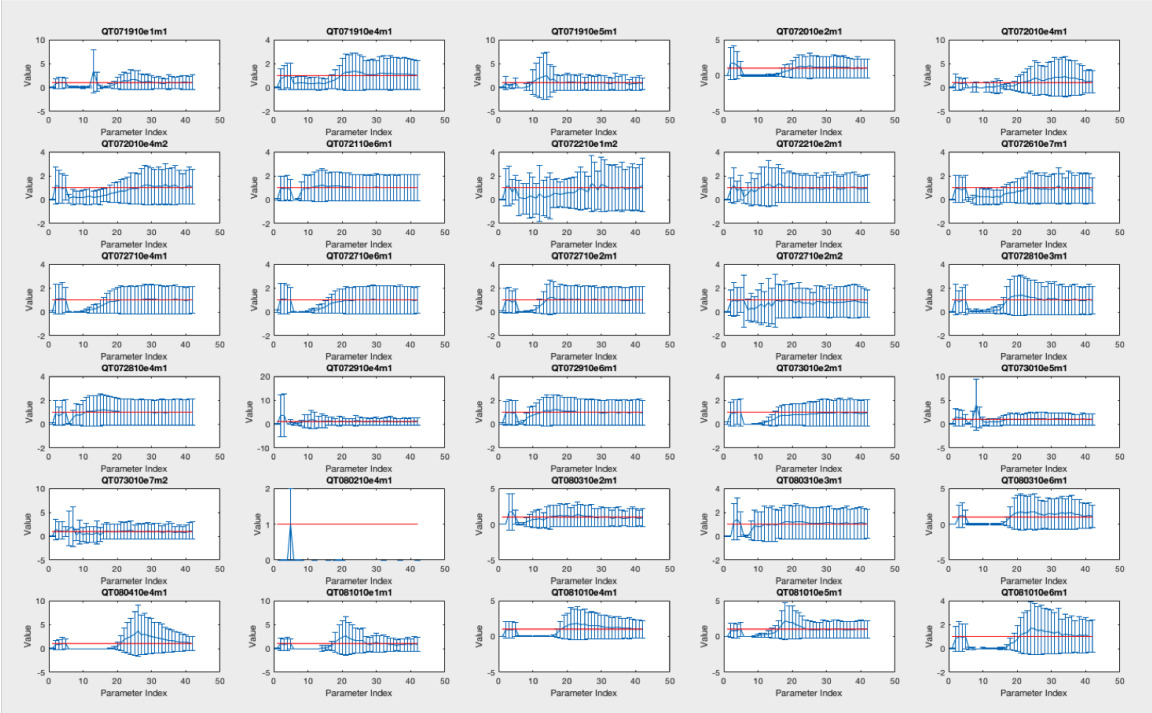


Figure 2.22: Parameter values and their confidence intervals calculated for each neuron.

groups of parameters from the 30 original parameters. This was performed for each of the 30 neurons. For each 29-parameter model, the average Akaike Information Criterion (AIC) for that model across all of the neurons was calculated. The model with the lowest AIC was used to seed the next iteration of this experiment. In other words, whichever parameter that model was missing was the parameter that was eliminated from the analysis. Figure 2.25a shows the average AIC values computed during each iteration of this experiment. As parameters were eliminated, the average AIC increased, indicating that the models with more parameters were better models for this data. Figure 2.25b shows the parameters used in each model as well as which parameter was eliminated during each iteration. This experiment was very costly in

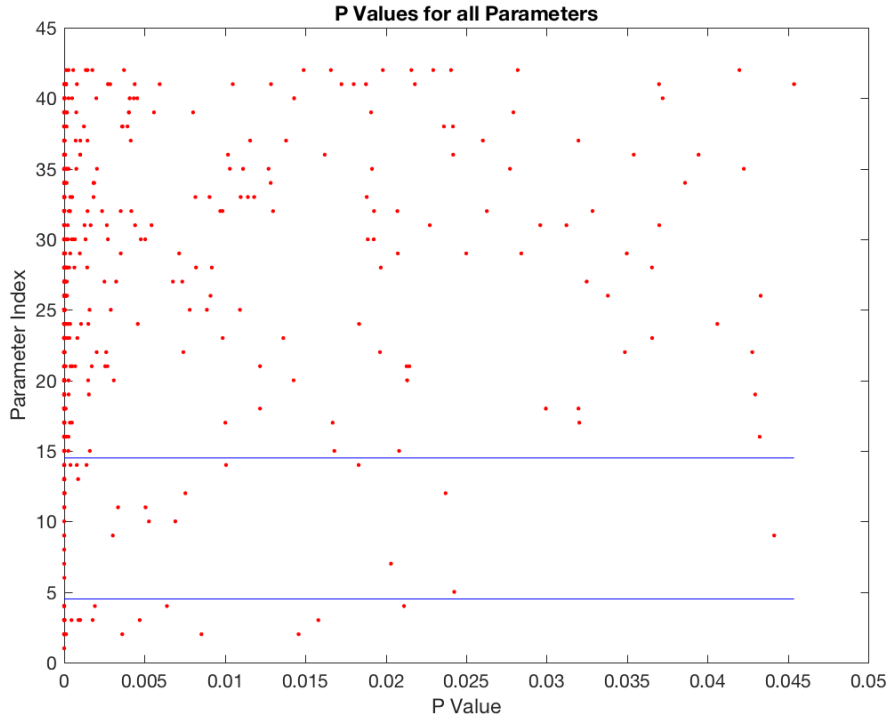
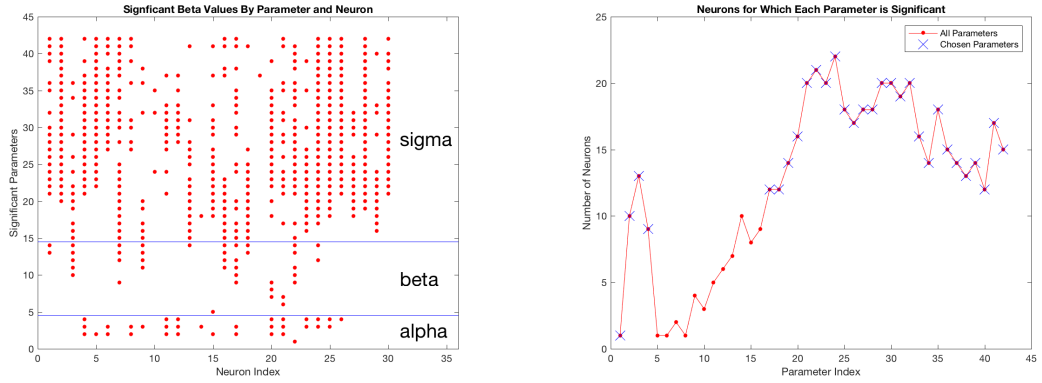


Figure 2.23: Parameter p-values for all 30 neurons. Lines drawn to indicate the α (parameters 1-4), β (parameters 5-14), and σ (parameters 15-42) parameters.

time due to the size of the covariate matrix and the number of neurons over which the AIC was averaged.

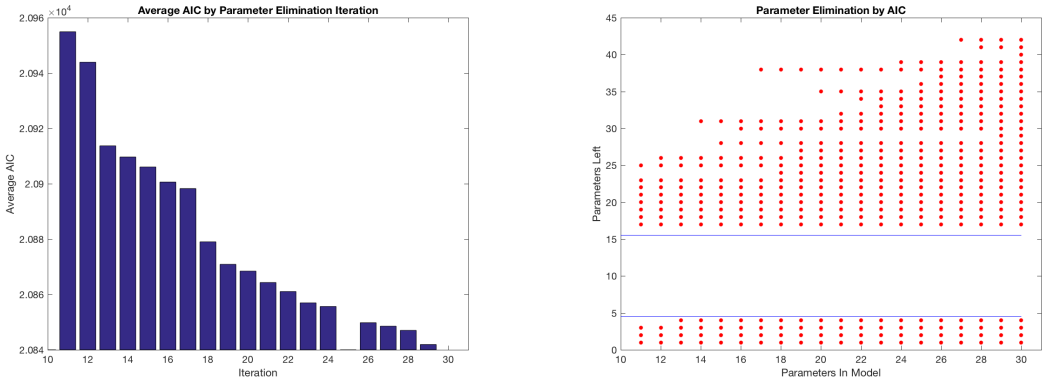
2.4.5.3 Clustering on Parameters

Because the nuclei for each of the 30 neurons used in this analysis was known, it was hypothesized that the parameters determined through the model could be clustered. Ideally these clusters would correspond to the neurons from each nucleus. K-means clustering was implemented on the 42 parameters. Figure 2.26 shows the results of this clustering, which indicates that the parameter clustering did not cluster



(a) (b)

Figure 2.24: (a) For each neuron, parameters that were significant were marked with a red dot. Blue lines divide the α (movie type), β (short term history), and σ (long term history) parameters. (b) Indicates the number of neurons for which each parameter was significant. Parameters marked with a blue X were used in the parameter elimination analysis described in 2.4.5.2.



(a) (b)

Figure 2.25: (a) Average AIC (across all 30 neurons) for each model. (b) Illustrates which parameters remained after each iteration of the elimination process.

the neurons by nuclei. Had the clustering matched the amygdala nuclei, all neurons within a specific nucleus would have been assigned the same "Nucleus Index" and that index would have been different from the index of other nuclei. In addition to

clustering on all 42 parameters, clustering was performed on just the 30 parameters chosen as a part of the parameter elimination experiment and also just on the four movie category parameters. None of these clustering metrics led to effective clustering to indicate amygdala nucleus type.

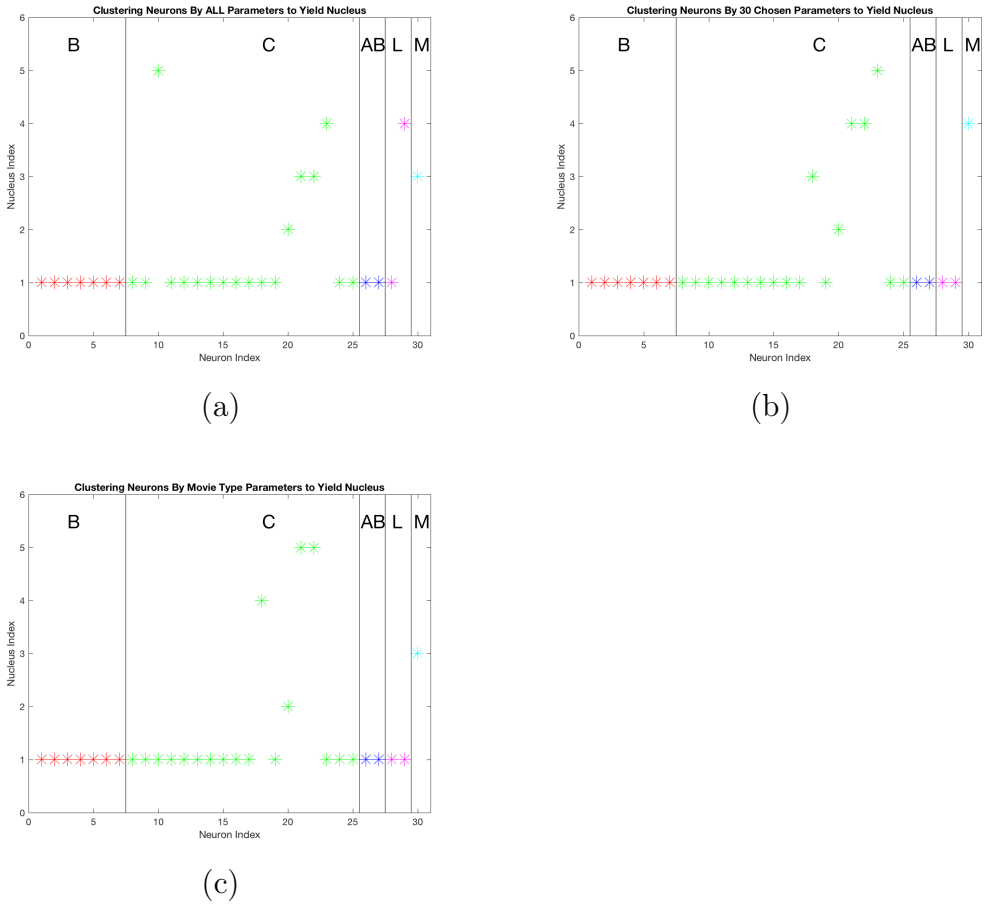


Figure 2.26: (a) Neurons were clustered using k-means clustering based upon the 42 parameters of the model. Neurons did not cluster based upon the nuclei of the amygdala from which they resided. (b) Neurons were clustered based the 30 parameters used in the parameter elimination experiment. (c) Neurons were clustered based off of only the α parameters.

2.5 Analysis of Bat Group Communication

Unrelated to the analysis of the amygdala's role in social processing, this section describes a tool created to facilitate the study of how bats use echolocation in groups, a primary method for their social interaction and communication.[107] Researchers interested in understanding bat communication run many experiments involving multiple bats interacting in a set environment. These experiments require much post-processing on the collected data to deduce from multiple microphones and cameras the locations of each bat and their individual echolocation calls throughout the experiment, before proceeding to answer any larger research questions. This section describes an interactive graphical user interface (GUI) built in MATLAB to facilitate this post-processing. Previously the existing GUI to assist with the processing of these trials required 20+ minutes per file, and required many mouse-clicks per call. The tool described here reduces the time post-processing time, while achieving an accuracy of $\sim$70% at best.

2.5.1 Data

This work was a collaboration with graduate student, Michaela Warnecke, in Professor Cynthia Moss's Comparative Neural Systems and Behavior Lab at Johns Hopkins University. All of the data discussed in this section was collected by members of Moss's group, and the tool discussed in this section was created to expedite the

analysis of their experiments. Details of the data and experiment can be found in their published work.[107]

2.5.2 Emission Time Call Identification

Like the existing GUI, this program to identify bat echolocation calls began by asking the user to select an audio file to analyze. Then the user was prompted to select a threshold above which peaks were considered echolocation calls, as seen in Figure 2.27. This selection was done for every channel used in the analysis. The user could adjust the number of channels in this algorithm (although only 24 of the 32 channels were actually used during the data collection process). The user threshold selection was retained for this program to simplify the process since it was already incorporated into the existing GUI. The call detection could also potentially be automated and accomplished using an algorithm without user input.

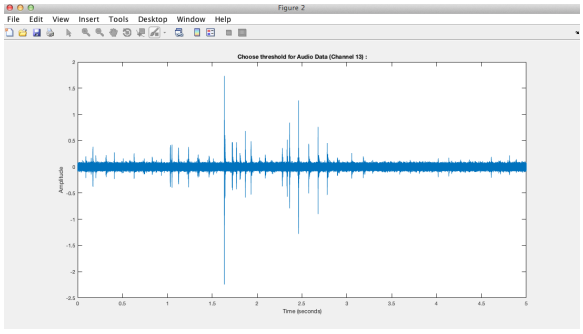


Figure 2.27: Screen shot of GUI through which the user selected a threshold above which peaks would be considered to be echolocation calls.

Once the thresholds were selected, calls were automatically identified using peak

detection and then the program launched the emission call identification program. The emission call identification program calculated the time each call would have been emitted had each bat emitted it, based upon the location of the bats and microphones. Then it determined which bat emitted the call, based on which bat had more proposed emission times that "lined up" in time. To accomplish this, first the program determined for each call, what time each call would have been emitted had Bat 1 emitted it or had Bat 2 emitted it, and those times were stored. Then the program went one by one through the calls in the baseline channel. It first found the call in each other channel closest to the current call in the baseline channel within a given window. If there were no calls in that window in the other channel then the program did not consider that other channel for this analysis. If there was a call, the program took those potential emission times for Bat 1 and Bat 2 for that call and added them to an array. After all other channels were cycled through, all of the potential emission times for Bat 1 and for Bat 2 were considered. The program calculated the variance of those calls for each bat and determined that the bat that emitted the call to be the bat whose emission times for this call had the least variance. Figure 2.28 shows the GUI developed to help visualize this process.

Once the bat that emitted each call was identified, the GUI would launch into another stage where the user could cycle through each call and check that it was been matched to the correct bat. A screen-shot of this GUI can be seen in Figure 2.29.

Unfortunately this program did not work as well as anticipated. This program did

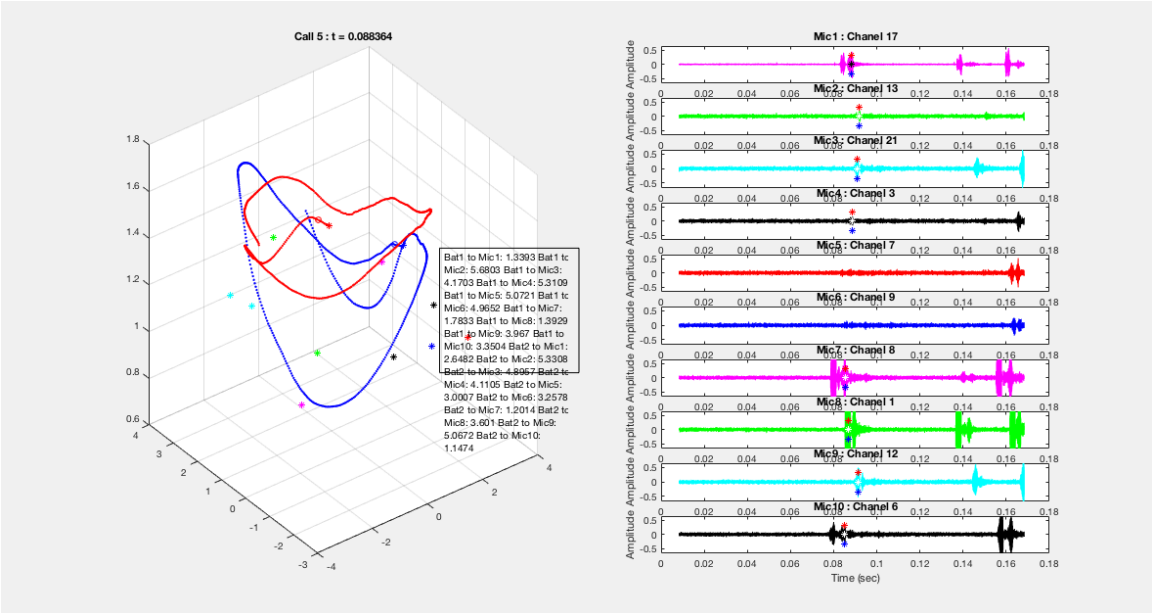


Figure 2.28: GUI used to visualize emission call identification function. The left shows the experiment setup. Red dots indicate the flight path of Bat 1 and blue dots indicate the flight path of Bat 2. Microphones are marked with a colored asterisk and the distances from each bat to each microphone are shown for debugging purposes. On the right, each channel's audio data surrounding the time of the current call in consideration is shown. A white asterisk indicates the call in each non-baseline channel considered to match to the call being considered in the baseline channel. The red asterisk indicates the time this call was emitted if Bat 1 emitted it, and the blue asterisk indicates the time this call was emitted if Bat 2 emitted it. These asterisks can be visually aligned to determine which bat actually emitted the call.

not work well because when the call threshold was determined, the program would often mark echos as calls and then identify the bat who emitted that echo. The program had no way to remove echos or any noise that were marked as calls.

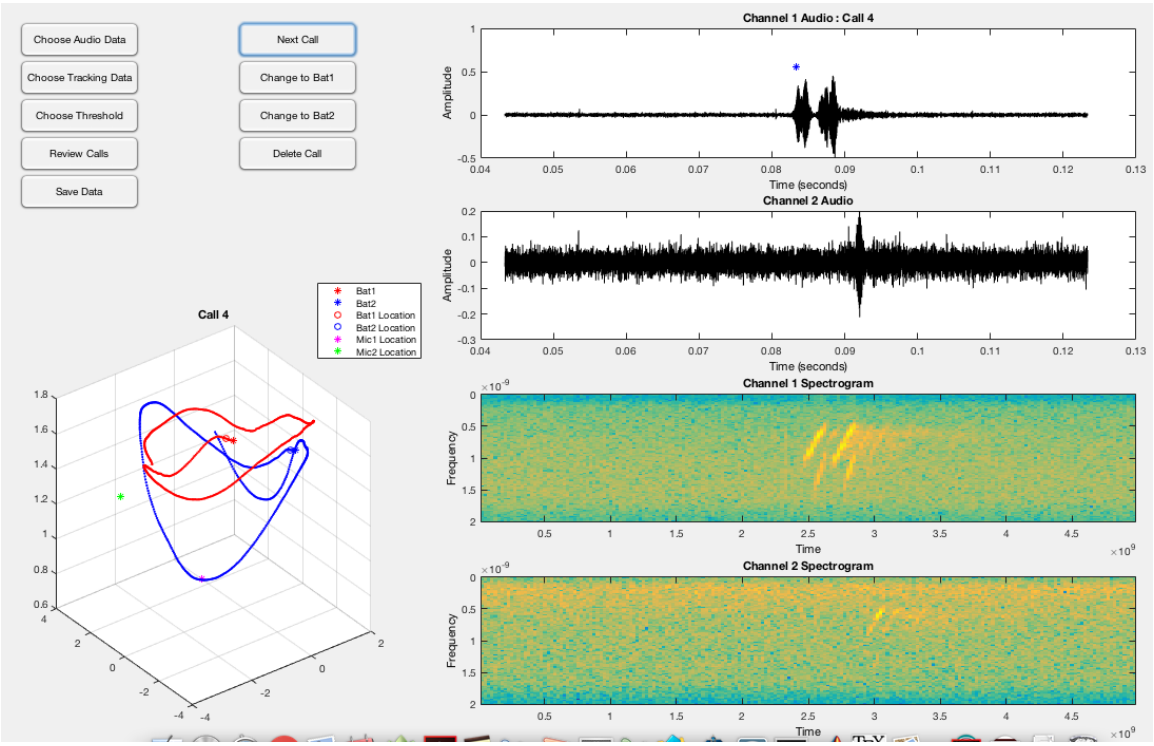


Figure 2.29: Post processing GUI allowed the user to check the calls the program automatically identified as being from a specific bat. The left plot shows each bat's trajectory throughout the five second trial. (Bat 1 is always red, Bat 2 is always blue). Microphones are also indicated. The top right shows the audio time series data of two of the channels and then bottom right plots show the spectrogram information for that same time period.

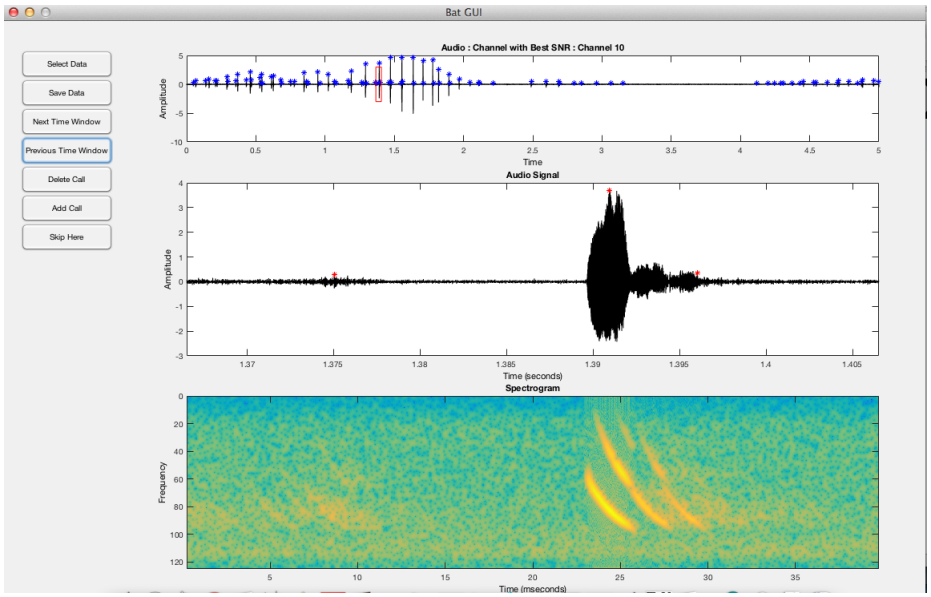
2.5.3 Improved Bat Echolocation Identification Program

To eliminate some of the issues of the initial program, a second program was created which first had the user go through two channels in detail so that only actual echolocation calls were marked as calls, and no noise or echos were selected in those channels. This was accomplished using a new GUI. Two versions of this GUI were created to provide different functionality and can be seen in Figure 2.30. Both GUIs

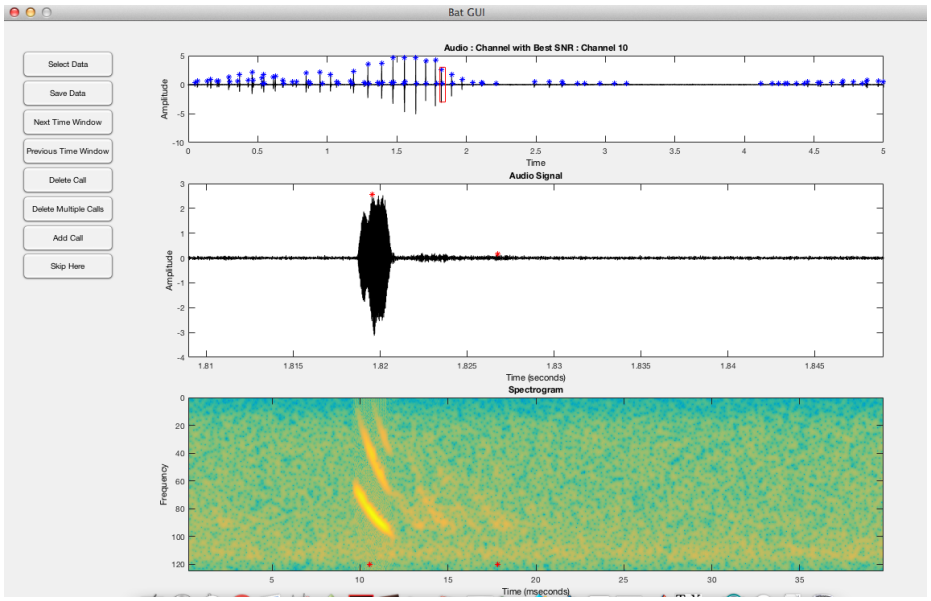
analyzed data from two channels, the channel with the best signal-to-noise ratio (SNR) and the channel corresponding to a microphone on the floor. If the channel corresponding to the microphone on the floor was the best SNR channel, then the program analyzed the channel corresponding to the microphone on the floor and then the channel with the second best SNR channel.

When going through these two baseline channels, like the initial program, first the user was asked to provide a threshold from which a first pass of calls are marked. The GUIs cycles through the channel audio data one time window (each window covers 0.04 seconds and windows overlap by 0.02 seconds) at a time. During each time window the user could add calls that were missed by the automated call identification process or delete calls that are not actual calls but instead noise or echos in the current time window. Once the entire audio signal was analyzed, the data was saved (or passed to another function) for further processing and call identification. To identify the calls, the same emission time call identification method that was described in Section 2.5.2 was called.

However, before the emission time call identification function was run to determine which bat emitted each call, the output from the pre-processing GUI was combined to make a "baseline" call set. The program would go through the calls in the second audio channel (the audio data from the floor microphone or second best SNR channel) and make sure that all of these calls were in the first channel call list by checking to make sure that there was a call in the first channel within a 0.1 second time window



(a)



(b)

Figure 2.30: Screen shots of the two pre-processing GUIs, each provided different functionality to the user.

of the call in the second channel. This final list of calls became the baseline for all future processing. If a call was not in this list, the program would not identify it

or match it to a bat. Changing the algorithm to not use the calls from these two channels as the baseline could improve the accuracy of the program.

Initially the peak time of each call was used in the subsequent processing, but the program was altered to instead calculate the start time for each call in the baseline list and then use call start times in all subsequent processing.

2.5.4 Time-Difference of Arrival Call Identification

Even after altering the initial pre-processing of the program to create a master call list from which to run the emission time call identification program, the accuracy was not optimal. Thus, another call identification algorithm was implemented using a time-difference of arrival (TDOA) algorithm[108] to determine which bat made which call. There were further issues with this implementation because of the data so a TDOA-inspired algorithm was developed.

The time difference of arrival (TDOA) inspired call identification algorithm worked similarly to the emission call identification algorithm in the sense that it cycled through the baseline call list. For each baseline call, the program took the time this call occurred and then calculated the distance from Bat 1 and the distance from Bat 2 to each of the microphones at the time of the call. The program then sorted the distances from each bat to each microphone to determine the order of microphones beginning with the microphone closest to the bat (i.e. the microphone that should first see the bat's call in its audio signal) to the microphone furthest from the bat

(i.e. the microphone that should see the bat's call last in its audio signal).

Next, the program considered the audio data in the other microphone channels. Like in the emission program, it first found the call in the other channels that was closest in time to the current baseline call within a given window. The program then computed all possible combinations of three microphone channels and considered the three calls seen in those channels. If the start times of those calls aligned with the distance from Bat 1 to those microphones then a tally for Bat 1 was increased by one. If the start times of those calls aligned with the distance from Bat 2 to those microphones then a tally for Bat 2 is increased by one. If the start times for the calls did not match either bat-microphone distance, nothing was done. This program assumed that the call should appear in the microphones in the same order as the distance between the microphone and the bat that made the call. After all groups of three microphones were analyzed, whichever bat had a higher tally was considered to have emitted the call. Figure 2.31 illustrates this process and provides visual confirmation to check that the program was performing correctly.

This algorithm was implemented as such because it did not require code to match calls between channels. A cross-correlation program was briefly implemented to do this, but it did not work well. Thus the previously described TDOA-inspired algorithm was implemented to provide more flexibility. If call matching was implemented so that the program was confident it was only analyzing the same call just seen in another channel, the program would likely work at a higher accuracy.

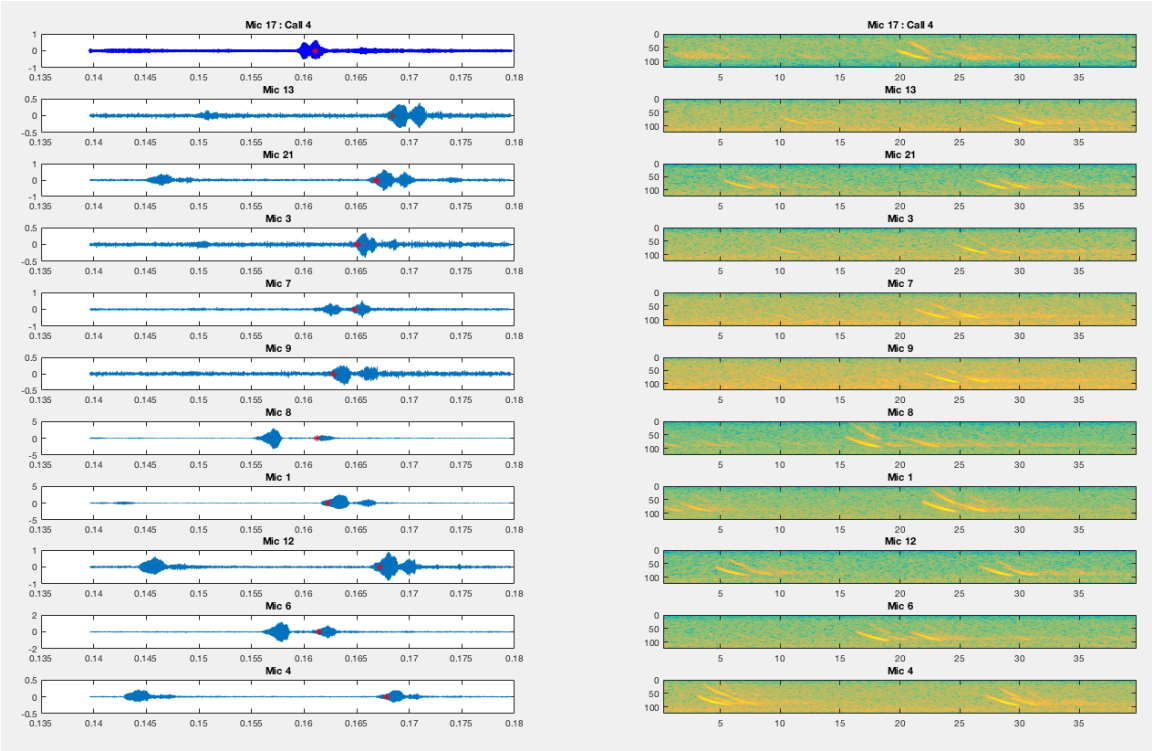


Figure 2.31: GUI used to visualize TDOA inspired call identification function. The left shows the audio data in each channel surrounding the current call. The current call from the baseline channel is shown in the top left plot and the call start time is marked with a red asterisk. The call that is closest in time to the baseline call, and thus considered during this analysis, is also marked with a red asterisk at its start time. The right side of the plot shows the spectrogram for each channel. When this plot was shown, the MATLAB terminal window displayed the distances between each bat and each microphone, as well as the classification of which bat made the current call. Using this GUI, one can verify that classification based upon the order in time that the calls appear in the audio channels as compared to the physical locations of the bats and the microphones.

2.5.5 Results

To determine the accuracy of these methods, four call files were marked by hand using the original, slow-to-use GUI to give a truth data set. The GUI output was considered to be truth for the analysis discussed in this section. To determine the

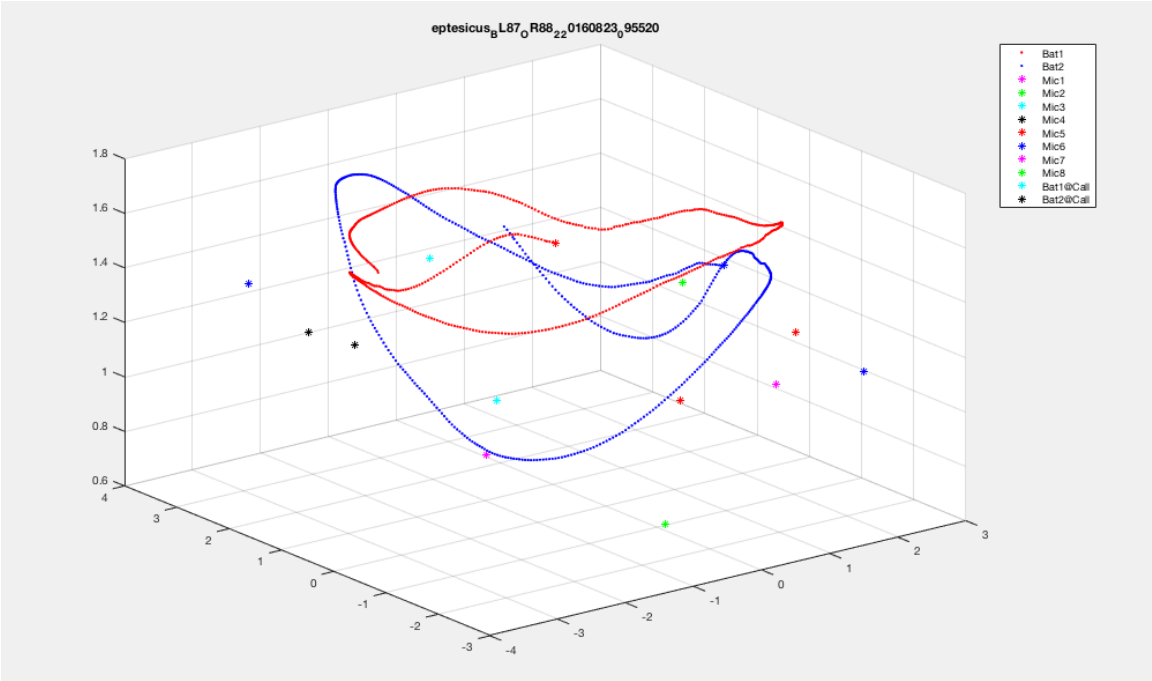


Figure 2.32: The program created this plot so that the user could visually verify that the bat and microphone locations were loaded correctly and made physical sense. This plot replicated the experimental setup.

accuracy of the program, first the calls output by the program were cycled through and the nearest call in the truth file to the current call in the program output was located within a given time window. Then the program went through the truth file and matched each call in the truth file to the closest call in the program's output file within a specified time window. If the same calls matched during both matching passes, then those calls were considered to be the same. Then if those calls were identified as coming from the same bat, the call was considered to have been correctly identified.

Figure 2.33 shows the plot displayed to indicate the results of the program as compared to the truth file. Calls marked with a red marker are calls that were

determined to be from Bat 1 (or calls from Bat 1 if marked in the truth file). Calls marked with a blue marker are calls that were determined to be from Bat 2 (or calls from Bat 2 if marked in the truth file). Calls marked with an "X" either were not matched to a call in the truth file because either they did not match to a corresponding call or the corresponding call indicated the other bat. Calls that were correctly identified and matched between the files are marked with the same marker to visually indicate that they are marking the same call. Using this GUI, the calculated accuracy of the program was verified.

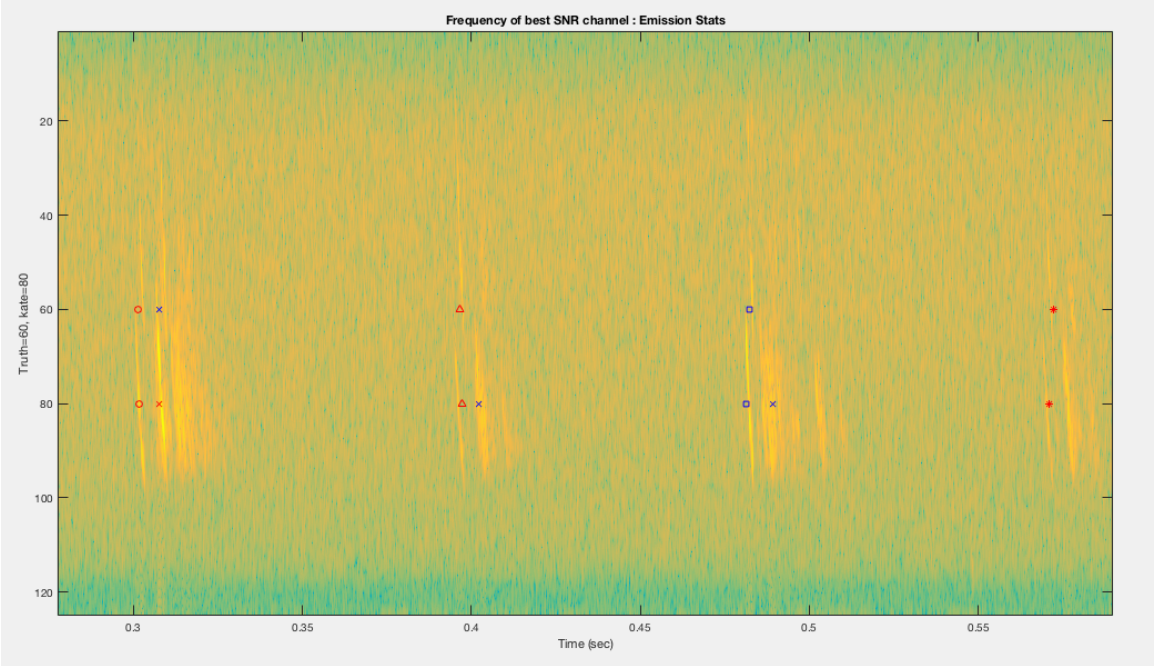


Figure 2.33: Plot shows identified calls in the truth file and calls identified by the TDOA-inspired program. This figure zooms in on a short time window so that individual calls can be seen. The actual figure displays the entire five second experiment window.

The results for both methods, as well as when the methods were used together

can be seen in Table 2.3. For this analysis 10 channels of the 32 were used. When the methods were used together it meant that a call was only marked as from Bat 1 if both the emission algorithm and TDOA-inspired algorithm marked it as from Bat 1. The same is true for Bat 2. Overall the best accuracy on a specific data set was seen when using the emission call program, and an accuracy of 68.82% was obtained. Nonetheless, overall accuracy was barely better than chance.

Experiment	Method	True Positive	False Positive	Accuracy
DataSet 2	Emission	47.83%	41.30%	49.62%
DataSet 3	Emission	71.11%	28.89%	68.82%
DataSet 4	Emission	43.18%	30.68%	50.00%
DataSet 2	TDOA	48.55%	42.75%	50.38%
DataSet 3	TDOA	56.67%	43.33%	54.84%
DataSet 4	TDOA	47.723%	26.14%	55.26%
DataSet 2	Emission + TDOA	63.10%	36.91%	39.85%
DataSet 3	Emission + TDOA	73.02%	26.98%	49.46%
DataSet 4	Emission + TDOA	66.00%	34.00%	43.42%

Table 2.3: True positives, false positives, and accuracy for the emission time call identification method, TDOA-inspired identification method, and using both methods at once.

2.5.6 Conclusions

One major problem with the way this program worked was that it would only be as accurate as the baseline channel. If the baseline channel was missing calls, there was no way for the program to find them. Specifically in a scenario where the two channels that made up the baseline call list were from the same side of the room, it is likely they would be missing calls made from the other side of the room.

Additionally if the baseline channel contained echos, then the final program output contained echos. Lastly, because the program did not "find" the current call in the other channels, it is likely that it would mistake calls and then "match" a call from one bat measured in one channel to a call from the other bat measured in a different channel. Adding in functionality to better find one call in another channel could help alleviate this problem.

Chapter 3

Neuromorphic Modeling of the Amygdala

The fundamental function of the amygdala is to discriminate between objects and events with positive and negative emotional significance. In primates, the amygdala also has social functions including supporting the discrimination and emotional evaluation of facial expressions.[65,109,110] Abnormal functioning of, or structural damage to the amygdala causes deficits in interpreting and recognizing facial expressions,[111,112] and it has been shown in humans that individuals with larger and more complex social networks have larger amygdala volumes.[113] The response properties of the neurons in the amygdala of humans and non-human primates suggest that processing faces engages a large number of neurons which respond selectively to face identity, facial expressions, and the direction of gaze and eye contact.[110,114–116] Neural activation in

the amygdala also correlates with the autonomic responses elicited by images of faces and facial expressions.[62,63] The recognition of facial expressions, their emotional evaluation, and the elaboration of emotional responses may emerge from different population of neurons within the amygdala. The amygdala is not anatomically homogeneous, rather the main amygdala nuclei have different cellular compositions, different connectivity, and sizes. Although neurons in the amygdala have been shown to respond to multiple task and stimulus-related variables,[57,117,118] a division of labor among the main nuclei of the amygdala is also empirically supported.[57] The model described in this chapter is based on the premise that different nuclei in the amygdala contribute to different components of the process that allow humans and non-human primates to communicate socially and emotionally with facial expressions.

3.1 Existing Models of Social and Emotional Processing

Understanding how the brain processes social and emotional stimuli is a complicated task, as previously explained in Chapter 2. The neural regions believed to be responsible for this processing, like the amygdala, lie deep in the brain which makes them particularly difficult to measure. The amygdala also receives inputs from and produces outputs to many different brain regions which makes isolating inputs and outputs difficult.[32,33]

Despite these challenges, there have been many models proposed to reproduce emotion and cognition both in amygdala-specific structures and more broadly. Some broad models, which are based on differing theories of emotion and cognition, have been shown to encapsulate effects including coping, planning, reinforcement learning, memory, and decision making.[119] Other broad models, take into account more biological plausibility. These models consider the neural processes needed for emotion,[120] or specifically how emotion, cognition, and mental states can be represented in the brain regions like the amygdala and prefrontal cortex.[121] Many of these implementations still remain broad and offer a general methodology instead of a specific implementation.

Most existing computational models that focus specifically on the amygdala, focus exclusively on fear conditioning,[122–125] conditioning more generally[126,127] or only consider a single type of emotional response.[128] Many of these models do not consider biological plausibility or only incorporate it to a limited extent due to computational complexity. Moreover these models do not take into account any sort of functional breakdown within the amygdala, also ignoring nuclei connectivity.

A key issue to be addressed by the model described in this chapter is the division of labor among the nuclei of the amygdala. Characterizing the specific functions carried out by these anatomically distinct nuclei is critical for a better understanding of the amygdala as a whole. Recent studies in mice have made great strides in highlighting nuclear-specific functions and operations in the amygdala.[129,130] Neuroscientists have

just started to explore the nuclear specializations in monkeys and humans, which are expected to differ greatly from those in rodents.[57,62,131–133] The model presented in this chapter also goes beyond existing models by incorporating multiple emotional states beyond simply fear conditioning. Additionally, the model is constructed in such a way that it could easily be expanded to include more complicated inputs, emotional states, and responses.

3.2 Modeling Methodology

3.2.1 Experiment and Data

To validate the amygdala model, the output patterns from neurons within the model were compared to the output spiking patterns from neurons recorded within amygdalae of adult male rhesus macaques (Macaca mulatta). Model neurons were simulated and measured under the same experimental conditions as those experienced by neurons recorded within the primate amygdala.[57] In the original experiment, monkeys were trained to fixate on a cue shown on a screen. Once the cue was shown the monkey had a two second window to begin its fixation on the cue. Once fixation began, the monkey had to maintain fixation on the cue for 120 milliseconds. After the 120 milliseconds of fixation, there was a random delay of 200-300 milliseconds before the image was displayed for 1.5 or 3 seconds. If the monkey did not begin fixation, did not maintain 120 milliseconds of fixation on the cue, or did not remain looking at

the image for the duration of its presentation the trial was considered to have failed and immediately ended. If it was a successful trial, after the image duration there was an inter-trial period of about three seconds, and then the next trial would begin.

Each image set contained human faces, random objects, landscapes, animals, food items, and abstract images. There were 681 unique images (383 monkey faces and 298 non-faces) partitioned into 59 unique stimulus sets. On average each set contained 13 images with 8 monkey faces and 5 non-faces. During each recording session 2 - 22 cells were recorded simultaneously.

To replicate this experiment using the model amygdala, fixation or cue presentation times were drawn from a normal probability distribution determined to model the actual experimental timing information. Once a trial began, a random variable drawn from a normal distribution with $\mu = 0.3047$ and $\sigma = 0.0937$ indicated the amount of time before fixation began. Fixation occurred for 0.295 seconds. There were 0.01 seconds between the end of the cue presentation and the image onset. Images were displayed for 3 seconds, followed by a 3 second inter-trial time. Although there was primate amygdala neural data collected using a 1.5 second image presentation time period, for simplicity, simulations were only run using the 3 second image presentation time because more neurons were recorded under that experimental setup. The number of trials to simulate was explored during parameter analysis and is discussed in Section 3.3.7, but ultimately a simulation time of 100 was chosen as the default number of cycles to simulate.

The previously existing primate amygdala neural data consisted of 490 neurons measured from within the basal nucleus, accessory basal nucleus, lateral nucleus, central nucleus, media nucleus, and anterior amygdaloid area. These neurons were recorded from within three monkeys over the course of 6 years (2004 - 2010) by members of the Katalin Gothard's group at the University of Arizona.

3.2.2 Neural Responses

There were four main categories of cell responses that the original cell analysis focused on.[57] The previous analysis explored how neurons in the basolateral (BL) and centromedial (CM) nuclei responded to the experiment, showing that the neurons of these two regions responded differently, indicating a division of labor within the overall nuclei structure. With this analysis in mind, we began analyzing the model's neuron's responses in a similar pattern.

Figure 3.1 illustrates the experimental setup and the four main neural responses considered in this work. Firstly, neurons sometimes exhibited a sharp increase or decrease in firing rate immediately after the onset of the cue, followed by a rapid return to the baseline firing rate. These responses were categorized as phasic excitatory or phasic inhibitory to the cue depending on if the change in firing rate was above or below the baseline firing rate, respectively. Neurons also sometimes exhibited a phasic excitatory or phasic inhibitory response immediately after the image onset. Additionally neurons sometimes exhibited a sustained increase or sustained decrease

in firing rate as compared to the baseline for the duration of the image onset. These responses were termed tonic excitatory and tonic inhibitory, respectively. The neurons sometimes exhibited a phasic excitatory or phasic inhibitory response to the image offset. The neurons could exhibit no response to the onset of the cue, onset of the image, image duration, and image offset. Additionally, neurons could exhibit any combination of responses to those four categories.

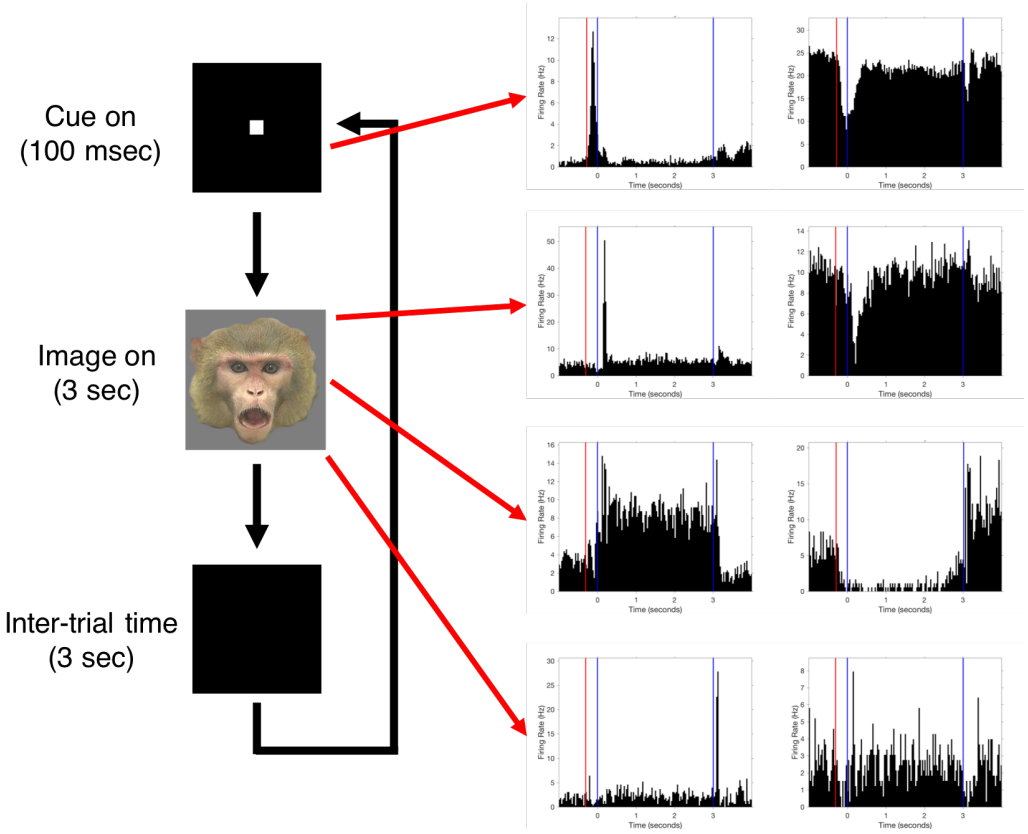


Figure 3.1: Illustrates the experiment setup as well as the types of neural responses to different parts of the experiment. The top row of neuron responses illustrate phasic responses to the cue. The row below those illustrate phasic responses to the image onset. The row below those illustrate tonic responses to the duration of the image. The bottom row illustrates phasic responses to the offset of the image.

3.2.3 Neural Information Encoding

Generally, this approach claims that each group of neurons represents some multi-dimensional vector $\mathbf{x}$, and that as the pattern of activity within that group changes over time, that value will also change over time, making it $\mathbf{x}(t)$. If one group of neurons represents $\mathbf{x}(t)$ and another represents $\mathbf{y}(t)$, then we may want to connect these populations such that $\mathbf{y} = f(\mathbf{x})$. That is, if the first group of neurons is firing with a particular pattern of activity that corresponds to $\mathbf{x}_1$, it's desirable that the synaptic connections to the second population will cause those neurons to fire with a pattern corresponding to $\mathbf{y}_1 = f(\mathbf{x}_1)$. If a set of connections can be found to make this work for the full range of $\mathbf{x}$ and $\mathbf{y}$ values, then the neural connections between the two groups can be said to implement the function $\mathbf{y} = f(\mathbf{x})$. This general methodology is known as the Neural Engineering Framework[28,29] (NEF), and is described in more detail in the following sections. The NEF was used to build the computational spiking neuron model of the amygdala described in this chapter.

3.2.3.1 Encoding Information in Neurons

If a group of neurons is to represent the multi-dimensional time-varying vector $\mathbf{x}(t)$, there must be some mapping from $\mathbf{x}$ to the input for each of the neurons within the group. For this work, we choose a simple population code based on the classic population representations observed in motor cortex by Georgopoulos.[134] Here, each neuron has a preferred direction vector in the space $\mathbf{x}$ is being represented. That

is, if $\mathbf{x}$ is three-dimensional, then for each neuron i we select a three-dimensional unit vector $\mathbf{e}_i$ and postulate that the somatic current flowing into the neuron at any moment in time should be based on the similarity between $\mathbf{x}$ and $\mathbf{e}_i$. Furthermore, to introduce heterogeneity into the population, each neuron has a scaling factor α_i and a background current β_i. This means that the total somatic current flowing into a neuron over time is $J_i(t) = \alpha_i \mathbf{e}_i \cdot \mathbf{x}(t) + \beta_i$. The current J_i can be treated as the input to any neuron model; here we use the standard Leaky Integrate-and-Fire (LIF) neuron model for simplicity. This approach has been shown to match well to a wide variety of neural firing patterns found throughout the mammalian brain, including V1,[135] M1,[136] and the pre-frontal cortex (PFC).[137]

This encoding process is also depicted in Figure 3.2. The first row shows the values ($\mathbf{x}(t)$ and $\mathbf{y}(t)$) that are to be encoded into the spiking patterns in the neural groups A and B, respectively. The second row shows the resulting spiking activity of eight neurons from each population. For demonstration purposes, we have chosen the $\mathbf{e}_i$ values for these eight neurons as follows. The first neuron is only sensitive to the first dimension in the value being represented (the blue line). Notice that it fires faster when the blue line is high and slows when the blue line is low and is completely insensitive to the orange line. This corresponds to $\mathbf{e}_1 = [1, 0]$. That is, the current flowing into the soma of the first neuron J_1 is as follows:

$$\begin{aligned} J_1 &= \alpha_1 \mathbf{e}_1 \cdot \mathbf{x}(t) + \beta_1 \\ &= \alpha_1 (e_{1,1} x_1(t) + e_{1,2} x_2(t)) + \beta_1 \\ &= \alpha_1 (1 x_1(t) + 0 x_2(t)) + \beta_1 \\ &= \alpha_1 x_1(t) + \beta_1 \end{aligned} \tag{3.1}$$

This will produce a large input current value whenever the first dimension of $\mathbf{x}$ is large and a small input current value whenever the first dimension of $\mathbf{x}$ is small.

Similarly, the second neuron in each population in Figure 3.2 fires more quickly when the first dimension is small and fires less quickly when it is large. This corresponds to $\mathbf{e}_2 = [-1, 0]$. The third and fourth neurons follow the same pattern as the first and second, except they are sensitive to the second dimension (orange line) rather than the first (blue line). These correspond to $\mathbf{e}_3 = [0, 1]$ and $\mathbf{e}_4 = [0, -1]$. The remaining fifth through eighth neurons are sensitive to both dimensions, corresponding to $\mathbf{e} = [\pm\frac{1}{\sqrt{2}}, \pm\frac{1}{\sqrt{2}}]$. In general, the $\mathbf{e}_i$ values are randomly chosen from the unit hypersphere. It should be noted that this is an example of the same sort of population coding using "preferred direction vectors"' that has been identified throughout the brain.[134]

Now that the spiking activity of neural population A is encoded in a time-varying vector $\mathbf{x}(t)$, in order perform additional operations on this data, the neurons from population A need to be connected to another population B. That is, population B should represent $\mathbf{y}(t)$, where $\mathbf{y}(t) = f(\mathbf{x}(t))$. For example, in Figure 3.2, the two

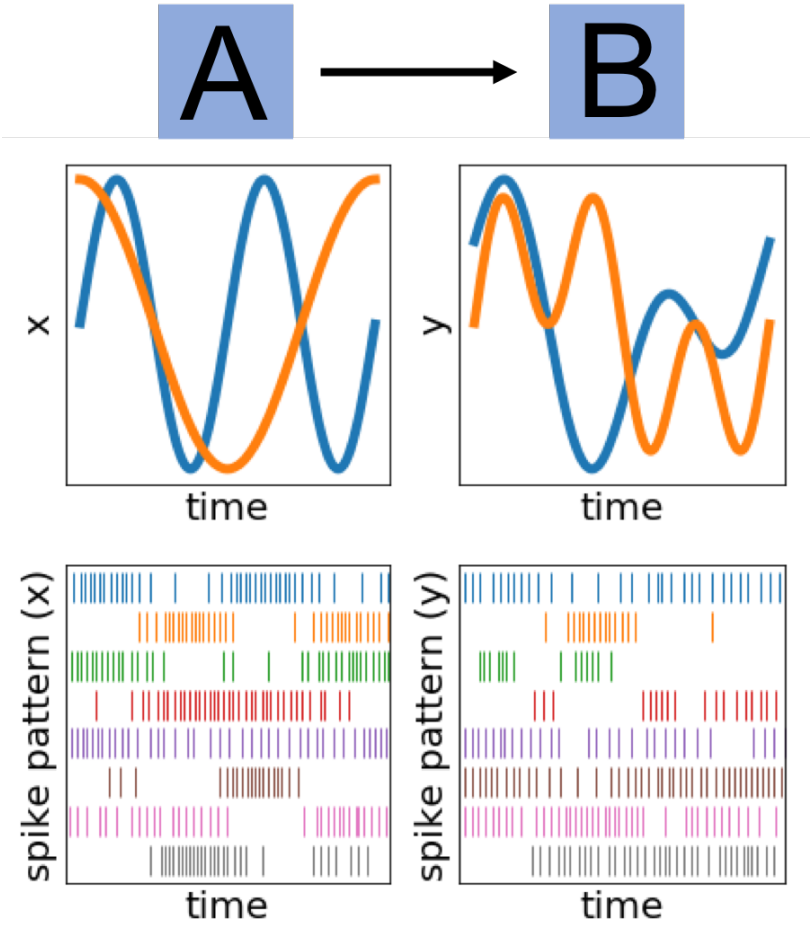


Figure 3.2: The method for defining connections between populations of neurons. Each population (A and B) encode some time-varying information (first row). That information is mapped to spikes (second row) by assigning each neuron a randomly generated tuning curve. The populations must then be connected such that if the population A is driven to create the given firing pattern, population B will produce a corresponding firing pattern. The connection weights that achieve this are found using least-squares minimization with the constraint that the connections from 80% of the neurons are positive while the remainder are negative (bottom row).

dimensions represented in population B are $(\mathbf{x}_1 + \mathbf{x}_2)/2$ and $\mathbf{x}_1\mathbf{x}_2$ (the mean and the product of the two dimensions represented in A).

The connection from neurons in population A to the neurons in population B is

accomplished by treating it as a least-squares minimization problem. The spiking activity produced by population A over time is the spike pattern produced by feeding the current $J_i(t) = \alpha_i \mathbf{e}_i \cdot \mathbf{x}(t) + \beta_i$ into each of the neurons in population A, which we will call $\mathbf{a}_A(t)$). The desired input current to population B is ($J_j(t) = \alpha_j \mathbf{e}_j \cdot \mathbf{y}(t) + \beta_j$). The connection weights ω_{ij} that minimize the difference between $\mathbf{a}_A(t)\omega$ and $\mathbf{J}_B(t)$ (i.e. minimize $(\mathbf{a}_A(t)\omega - \mathbf{J}_B(t))^2$) can then be found.

3.2.3.2 Nengo

Nengo is a Python library that implements the NEF and allows users to build neural models through a graphical user interface or by scripting.[138] It provides an interface for model manipulation and simulation, allowing users to observe neural populations interacting during simulation. Nengo has been used to model many aspects of cognition, including but not limited to, visual attention,[139] rodent navigation,[140] and reinforcement learning.[141] It has also been used to create Spaun, a functional brain model that uses 2.5 million neurons to perform eight different cognitive tasks.[142,143]

Figure 3.3 illustrates the connections between two neural populations, A and B, as described in the previous section. When using Nengo, after defining the neural populations and the functions performed through population connections, the NEF will calculate the appropriate encoding and decoding weight matrices to approximate the desired transformations on the information represented in a distributed fashion across each neural population. Figure 3.3 and Figure 3.2 both illustrate connections

between two Nengo neural populations. Specifically, Figure 3.2 emphasizes the population encoding aspect and Figure 3.3 emphasizes the encoder and decoder weight connections.

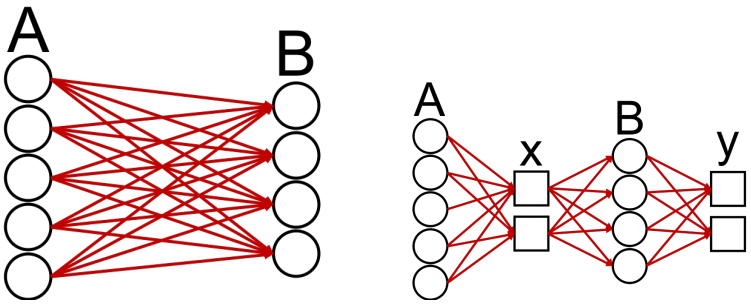


Figure 3.3: Illustration of basic Nengo neural population connectivity. (a) Connections are made between the neural population A and neural population B. (b) Each connection between two neural populations involves the multiplication of decoding weights and encoding weights. These weights are calculated by the NEF and Nengo using least-squares minimization to produce the weights necessary to calculate any function. This example implements an identify function. As illustrated, population A's activity corresponds to a distributed representation of $\mathbf{x}$, which can be decoded by multiplying the output of A by the decoding weights. $\mathbf{x}$ can then be multiplied by encoding weights to provide the input to population B. Population B's activity corresponds to a distributed representation of $\mathbf{y}$, which can be decoded by multiplying the output of B by the corresponding decoding weights. This figure was adapted from the author's previous work.[144]

3.3 Computational Model of the Amygdala

The model described here consisted of three amygdala nuclei, including a basal, lateral, and central nucleus. In this model the basal nucleus combined the basal and accessory basal nuclei, while the central nucleus combined the central nucleus, medial

nucleus, and anterior amygdaloid area. The model contained 37,000 neurons, where the amygdala portion of the model consisted of 12,000 neurons. To define such a large model, a general method for mapping a high-level description of the desired behavior into a low-level detailed model was used as described in Section 3.2.3.

3.3.1 Model Definition

Figure 3.4 shows both an anatomical depiction of the amygdala and a color-coordinated depiction of the model highlighting information flow through the system. For this model, a simplified visual processing system was constructed consisting of two neural populations, V1 and IT. Images were passed into V1 as input. V1 functioned as a traditional artificial neural network and was trained through gradient descent to output the emotional state of the monkey in the image (threatening, neural, fear grimace, or lip smack) and the gaze direction (directed or averted). A fear-grimace facial expression indicated a non-threatening state to a social partner. A lip smack facial expression indicated submission and low social status to a social partner.

In this model, the neural firing patterns within V1 represented a five dimensional value consisting of the input emotional state and gaze direction. That five dimensional value was then passed to IT as input along with a two dimensional value that indicated the presence or absence of a cue and the presence or absence of an image. IT aggregated those two signals and then passed the seven dimensional value to the lateral nucleus of the amygdala. The lateral nucleus's neural firing pattern represented

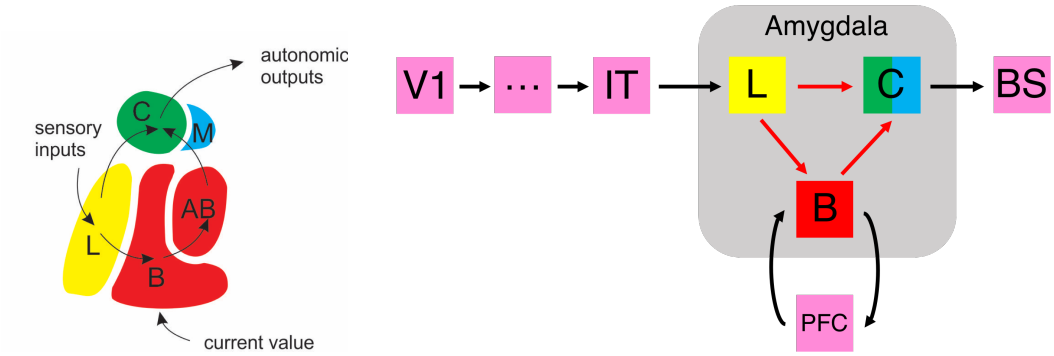


Figure 3.4: (a) Anatomical depiction of information flow through the amygdala nuclei. (b) Depiction of information flow through computational amygdala model.

the same seven dimensional value as IT.

The lateral nucleus passed information to the basal nucleus. That connection transformed the seven dimensional input value into a two dimensional emotional representation. There are many ways to represent the dimensionality of emotional responses, but this model used a two dimensional representation where one dimension represented valence and the other represented arousal. This representation is a simplified version of a common three dimensional emotional representation using evaluation (good versus bad), potency (powerful versus powerless), and activity (liveliness versus torpidity).[145] The two dimensional value represented by the basal nucleus was a combination of the information passed to it by the lateral nucleus and the information passed to it from a prefrontal cortex (PFC) neural population. The PFC stored the previous emotional state so that the current emotional state calculated by the basal nucleus was a combination of the current input and a previous state. A lip smack input expression elicited a response of medium positive valence and low arousal. A

neutral input expression elicited a response of neutral valence and no arousal. A fear grimace input expression elicited a response of high positive valence and high arousal. A threatening input expression elicited a response of high negative valence and high arousal.

Next, the basal nucleus passed information to a central nucleus, a winner-take-all circuit that determined which one of the four emotional responses that was then projected to the rest of the body, or as in Figure 3.4, a brain stem neural population. Lastly, there was also a connection from the lateral to central nucleus which only sent information that required an immediate fast responses, such as a threatening input. A summary of input images, the valence and arousal they elicited, and the corresponding response state are shown in Figure 3.5. Adverted and direct gaze resulted in the same determined value, although the arousal response tended to be larger when the gaze was direct.

3.3.2 Model Build and Simulation

There are two steps to execute a model using Nengo. First the model was built, i.e., the process of solving the least-squares minimization to determine all encoder and decoder weights to map the desired functions to neural population connections. These calculations depended on the number of neurons in the model as well as the complexity of neural population parameters. Once built, the model was simulated for a predetermined amount of time. If the model was small enough, it could be

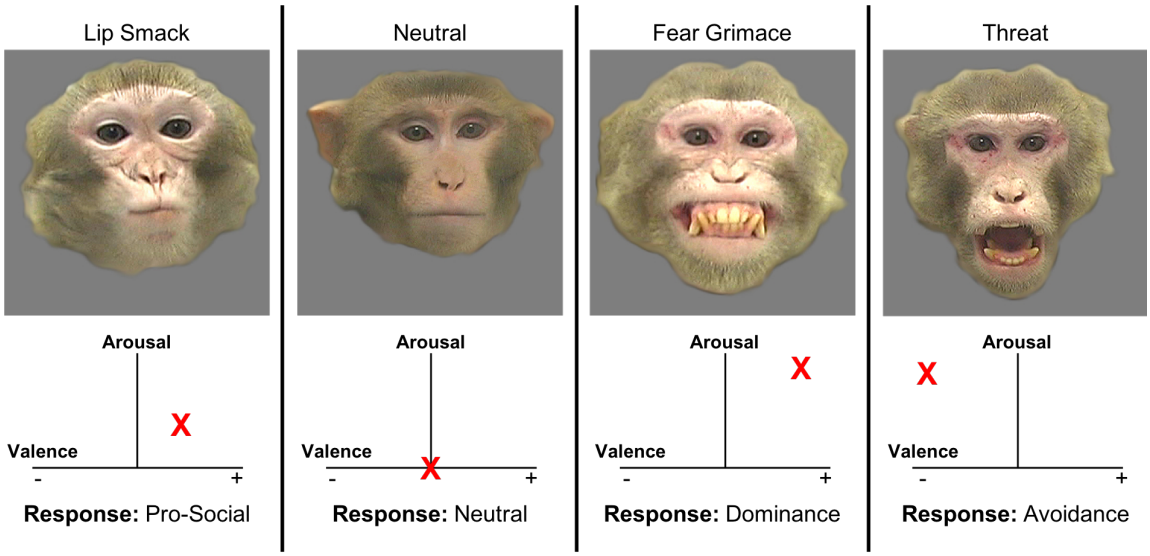


Figure 3.5: There are four categories of emotional expressions expressed by monkeys in the input image set. A lip smack expression elicited a response of medium positive valence and low arousal, resulting in a pro-social response by the viewer monkey's amygdala. A neutral expression elicited a response of neutral valence and no arousal, resulting in a neutral response by the viewer monkey's amygdala. A fear grimace expression elicited a response of high positive valence and high arousal, resulting in a dominant response by the viewer monkey's amygdala. A threatening expression elicited a response of high negative valence and high arousal, resulting in an avoidance response by the viewer monkey's amygdala.

rendered in a reasonable amount of time and viewed using a graphical user interface (GUI) provided by Nengo. If models were coded within the GUI, neural populations were connected in the GUI in real-time as they were described in code. The Nengo GUI is useful for visual inspection of a coded model, as well as for watching model simulations. A screen shot of the amygdala model in the Nengo GUI can be seen in Figure 3.6. To make this video, the amygdala model was reduced by a factor of 1000, and neuron firing rates were increased by a factor of two so that the video would render in a reasonable amount of time and execute without error.

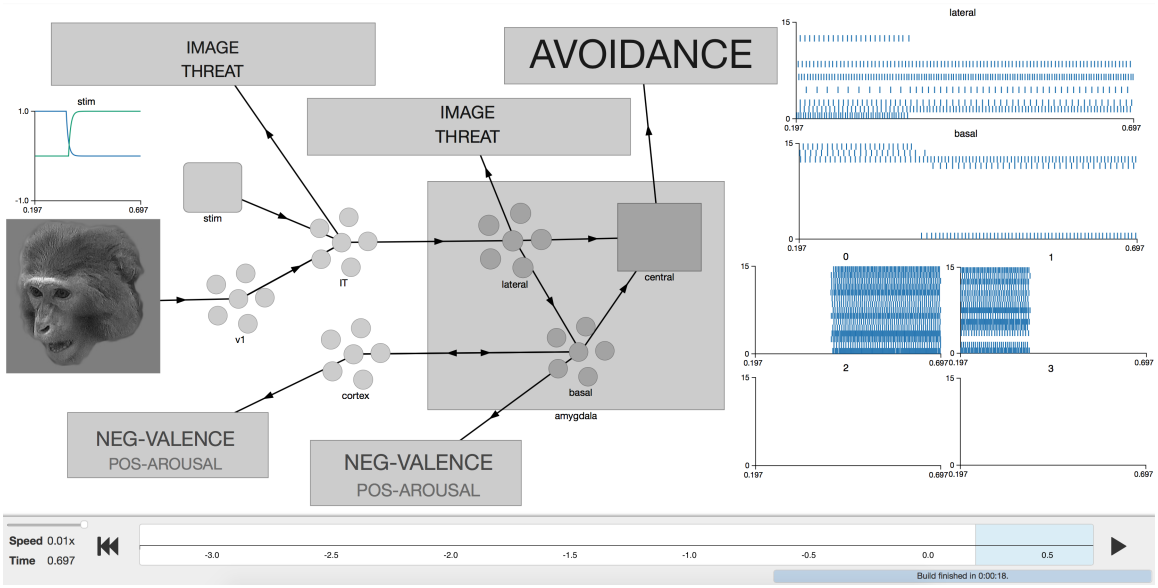


Figure 3.6: Nengo provides a graphical user interface (GUI) for visualization of Nengo simulations. Above illustrates the visual simulation of the amygdala mode in Nengo. All neural populations have word clouds illustrating their represented states. The darkness of the word indicates the strength of that representation. On the left 15 neurons from the lateral, basal, and four winner-take-all states within the central nucleus are shown. As the model executed in this GUI, the changes in firing rates and represented states in response to the changing input was illustrated.

3.3.3 Model Output

At the conclusion of the model simulation all relevant information was saved to a file. This included spiking information for all neurons within the model, model inputs, and model parameters. This data was then converted from Python to a MATLAB data structure and processed into the same form as the data and experiment parameters recorded in primate amygdalae.

3.3.4 Automatic Classification of Neural Responses

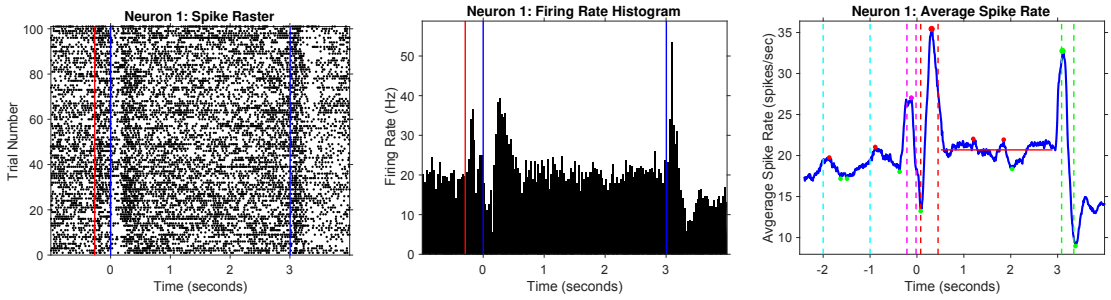
To compare the model outputs to the neuron responses recorded in the primate amygdala, each neuron was classified based on its firing responses throughout the experiment. As described in Section 3.2.2, neurons were identified as having either a phasic response to the onset of the cue, onset of the image, or the offset of the image. Neurons could also exhibit a tonic response throughout the duration of the image presentation. Neurons were classified to have an excitatory, inhibitory, or no response to each of these categories, resulting in 81 unique classifications of response. Responses were considered to be phasic to the cue if there was a sharp increase or decrease in firing rate 80 to 275 milliseconds after the cue came on, followed by a return to baseline firing rate. Responses were considered to be phasic to the image onset if there was a sharp increase or decrease in firing rate 80 to 450 milliseconds after the image came on, followed by a return to baseline firing rate. Responses were considered to be phasic to the image offset if there was a sharp increase or decrease in firing rate 80 to 274 milliseconds after the image turned off, followed by a return to baseline firing rate. Tonic responses were considered 450 milliseconds after the image came on until the end of the image presentation and corresponded to an increase or decrease in firing rate during that duration.

To automate this process, a classification algorithm was developed in MATLAB to classify each neuron into these categories. To do this the spikes from each trial were aligned relative to the image onset time and then aggregated and averaged across 0.2

second time windows with 0.1 seconds of overlap. Significant peaks and dips relative to the baseline firing rate were identified, and if they fell within the corresponding time window the neuron was considered to have a phasic response. A two sample t-test was used to determine if the neuron exhibited a tonic response to the image presentation by comparing the average firing rates during the image presentation and during the baseline. A visualization of this process is illustrated in Figure 3.7. This classification algorithm was developed by dividing the labeled primate neural data into train, development, and test sets and developing the classification algorithm on the primate neural data. Running this algorithm on the test data set achieved a classification accuracy of 84.3% for phasic responses to the cue, 78.1% for phasic responses to the image onset, 72.3% for phasic responses to the image offset, 79.9% for tonic responses, and an overall accuracy of 40.0%. To label the data for development of this classifier, process all primate neural cell firing responses were visually inspected and categorized with guidance from neuroscientist collaborators in Professor Gothard's group.

3.3.5 Primate and Model Amygdala Comparison

By observing the model executing in the Nengo GUI, one can verify the accuracy of the nuclei functionality and represented data through spot checks. However, this output cannot be validated using primate neural data since that functionality and representation is precisely the unknown aspects this model attempts to understand.



(a) (b) (c)

Figure 3.7: Visualization of the classification process. (a) Spike raster of neuron responses to individual trials. Red line indicates the onset of the cue. The left blue line indicates the onset of the image. The right blue line indicates the offset of the image. (b) Histogram indicating the firing rate across all trials. (c) Classification visualization performed on average spike rate signal calculated from the individual spikes across all trials of the experiment. Signal between the cyan dotted lines indicates the baseline period. Magenta dotted lines surround the time window for a phasic response to the cue. Red dotted lines surround the time window for a phasic response to the image onset. Green dotted lines surround the time period for a phasic response to the image offset. Horizontal red line indicates time period considered when determining tonic responses to the image duration.

Instead the neuron firing responses measured from the model were compared to that of the previously measured primate amygdala neurons. Figure 3.8 illustrates three neurons from the model and three neurons measured in the primate amygdala that exhibit the same firing responses to the experiment. In addition to exhibiting the same firing response, they exhibit similar firing rates. An initial pass through the model data proved promising for finding many cases of overlap.

After individual examination of neuron response plots, methods were developed to analyze the similarity of responses on a population level. Figure 3.9 shows the percentages of neurons within the primate amygdala and model that exhibit each of the 81 unique responses to the experiment. The four categories of responses (phasic to

the cue, phasic to the image on, phasic to the image off, and tonic during the duration of the image) combined with the three responses per category (excitatory, inhibitory, or no response) result in 81 unique responses to the experiment. Because there were different numbers of neurons recorded in each nuclei and each nuclei contains a different number of neurons, Figure 3.10 shows the same distribution calculation but calculated by nuclei. For example, many fewer neurons were recorded in the lateral nucleus of the primate amygdala because this nuclei is sparsely connected and neurons tend to fire at a lower firing rate making neural recordings more difficult to produce. In both of these figures, it is noted that the majority of neural responses were replicated in the model amygdala at the same percentage as measured in the primate amygdala.

3.3.6 Parameter Variation Analysis

To further quantify how well each of the model responses matched those of the primate amygdala, as well as to help quantify the accuracy of models with different parameters, a Hellinger distance was calculated. The Hellinger distance is used to quantify the similarity between two probability distributions. For two discrete probability distributions $P = (p_1, ...p_k)$ and $Q = (q_1, ...q_k)$ the Hellinger distance is defined as:

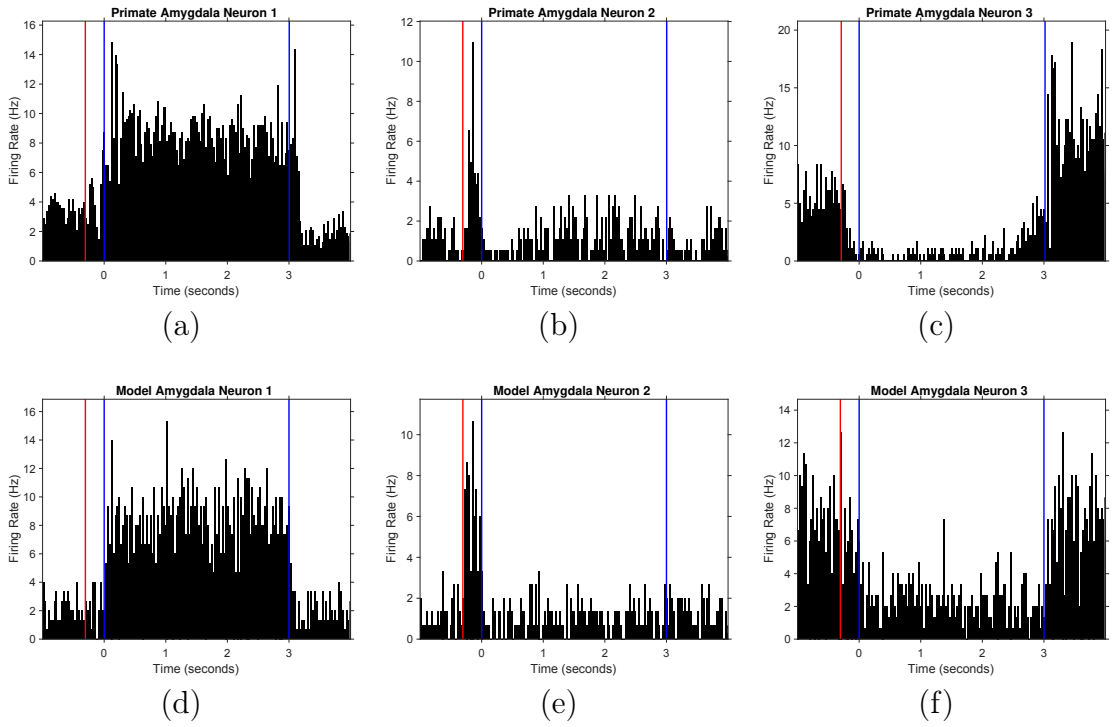


Figure 3.8: Three pairs of neuron responses to the experiment are shown, each containing the response from a primate amygdala neuron, and the response from a neuron from the amygdala model. These are three of many examples where model cell responses match those measured from primate amygdala neurons. (a) Primate amygdala neuron with tonic excitatory response to the image. (b) Primate amygdala neuron with phasic excitatory response to the cue. (c) Primate amygdala neuron with tonic inhibitory response to the image. (d) Model neuron with tonic excitatory response to the image. (e) Model neuron with phasic excitatory response to the cue. (f) Model neuron with tonic inhibitory response to the image. Like previous figures, the red line indicates the onset of the cue, the blue lines indicate the onset and offset of the image.

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2} \tag{3.2}$$

Using this distance the difference in the model neuron response distribution compared to primate amygdala neuronal response was quantified. This was particularly

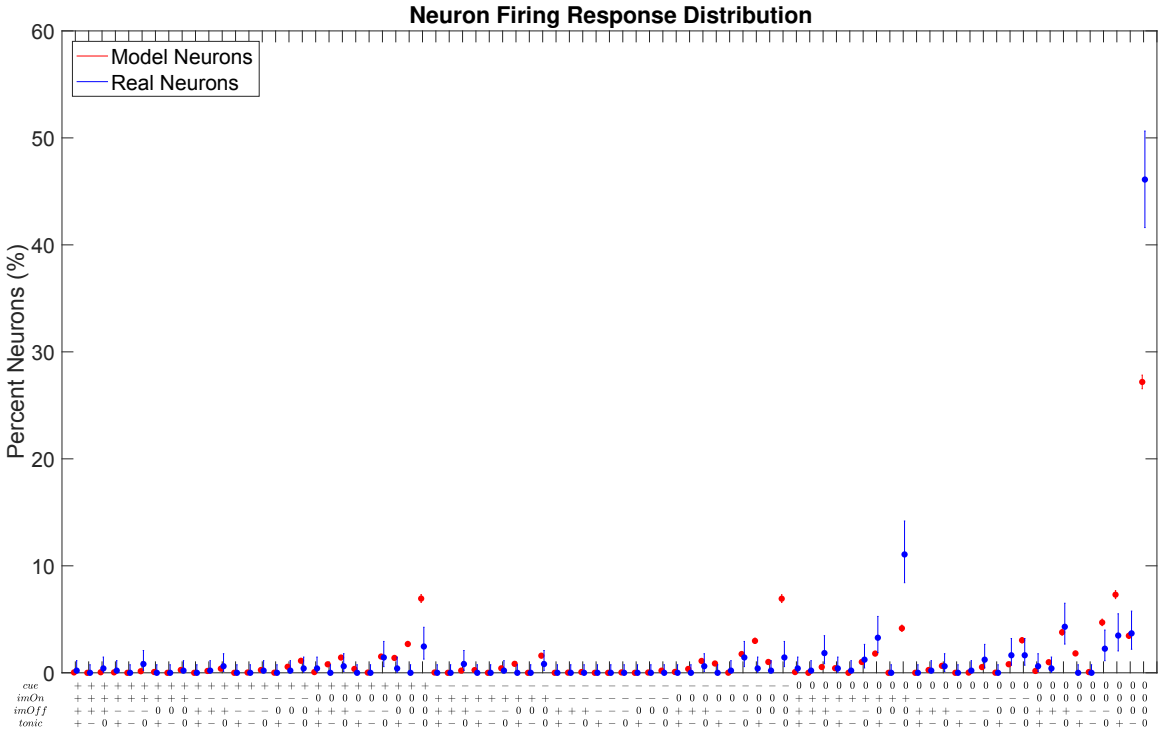


Figure 3.9: For each of the 81 unique responses to the experiment, the percentage of neurons within the population that exhibit that response is indicated. This value is shown for both the model neurons and primate neurons, with confidence bounds. The 81 unique responses result from the fact that neurons can have no response, an excitatory response, or an inhibitory response to the cue onset, image onset, image off, and duration of the image presentation. These response categories are not mutually exclusive. This plot contains data from 18,785 model neurons and 488 primate amygdala neurons.

useful for comparing models during parameter exploration. Originally a Kullback-Leibler (KL) divergence was to be used to quantify the differences in models. The KL divergence is a measure of how one probability distribution is different from a second, reference probability distribution. However the calculation to determine the KL divergence assumes all values of the distribution occur at least once, which is false for both the model and primate amygdala data. Therefore a Hellinger distance was

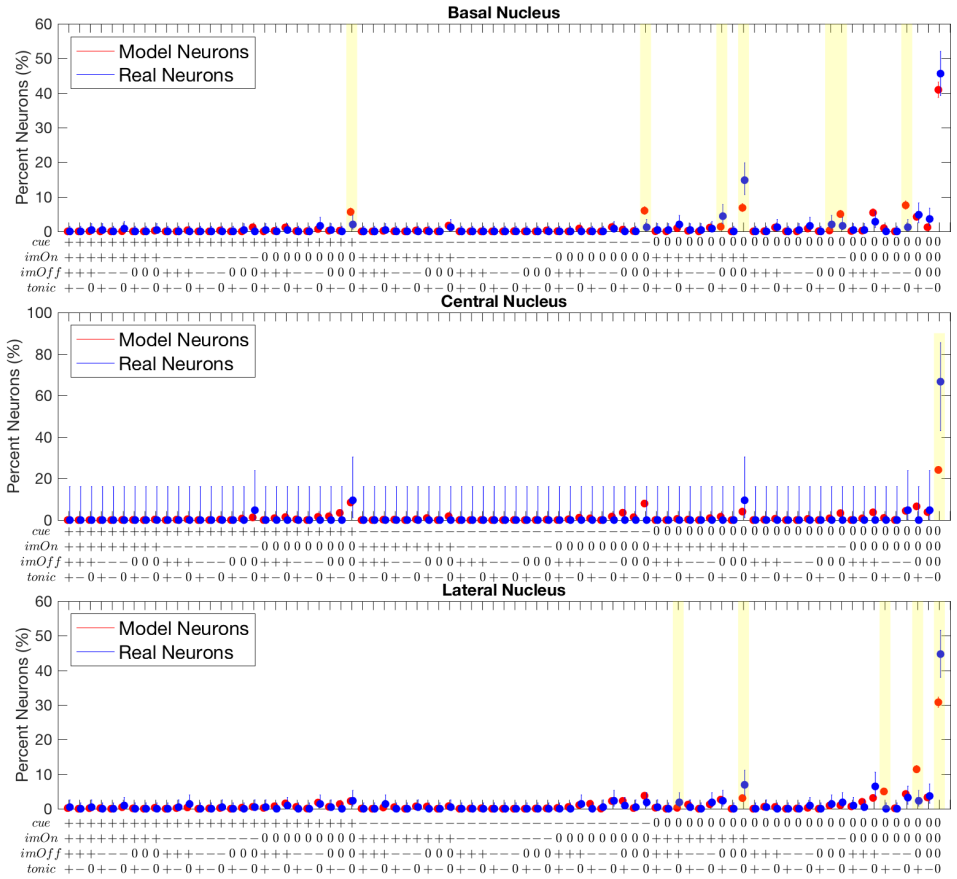


Figure 3.10: For each of the 81 unique responses to the experiment, the percentage of neurons within the population that exhibit that response is indicated by nuclei. When simulating the model all neurons are recorded throughout the experiment, but in vivo the recordings are not collected uniformly across the different nuclei. By comparing responses by nuclei, the model can be more accurately compared to the primate neural data. This plot contains data from 1,920 model cells and 250 primate cells from the basal nucleus, 13,153 model cells and 21 primate cells from the lateral nucleus, and 3,712 model cells and 217 primate cells from the central nucleus.

used instead.

3.3.6.1 Neuron Firing Rate Distribution

The first model parameter explored compared the neuron firing rate distribution from the model to that from the primate amygdala neurons. Nengo assumes a default uniform distribution with maximum rates from 200 - 400 Hz when building a model. The range 200 - 400 Hz is much higher than those observed from the neurons measured within the primate amygdala so the model neuron firing rate distribution was lowered accordingly.

Figure 3.11 shows the average maximum firing rate distribution for the primate amygdala neurons, fitted to lognormal, gamma, exponential, and normal distributions. A bin width of 0.5 seconds was used to generate the histogram upon which the distributions were based. The fitted distribution parameters were:

- **Lognormal** $\mu = 3.109$, $\sigma = 0.719$
- **Gamma** $k = 1.984$, $\theta = 14.826$
- **Exponential** $\lambda = 29.412$
- **Normal** $\mu = 29.412$, $\sigma = 24.636$

A lognormal distribution was selected to generate the neuron firing rates for the Nengo amygdala models. However, by decreasing the firing rates of the model neurons, the number of neurons had to be increased to maintain a similar level of accuracy in the decoded representation. This is discussed in the next section.

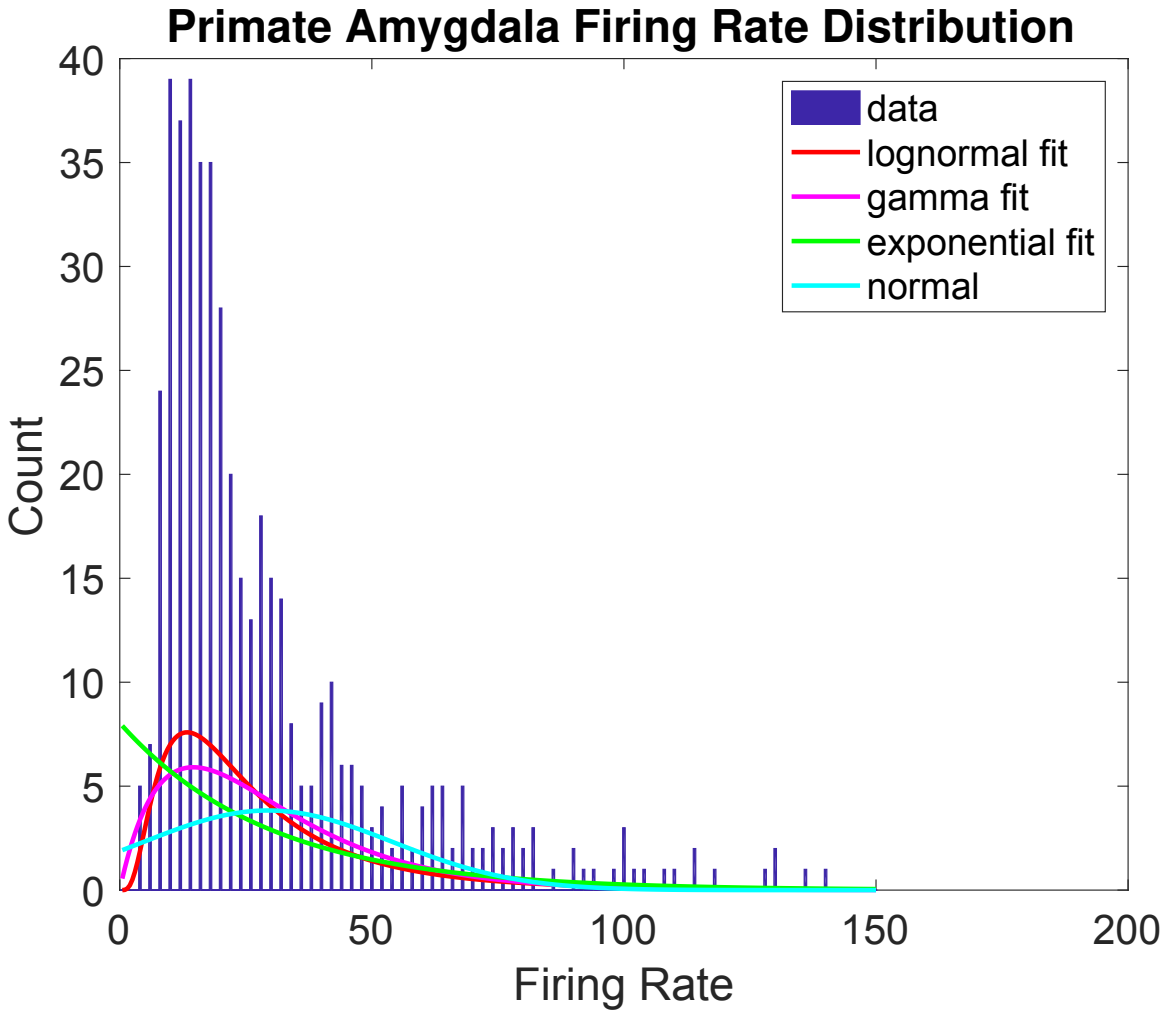


Figure 3.11: Plot illustrates the distribution of maximum firing rates for the primate amygdala neurons, with the distribution fitted to lognormal, gamma, exponential, and Gaussian distributions. A lognormal distribution was chosen to generate the distribution of firing rates for the Nengo amygdala models.

3.3.6.2 Nuclei Size

After determining the distribution of firing rates measured from the primate amygdala, the model neurons' firing rate distribution was decreased and coded to match. However, to maintain an accurate representation, the number of neurons per nuclei had to be increased. Figure 3.12 shows how the Hellinger distance increases as the

model size grows, as well as the rate at which the model size grows as a function of the neurons per dimension represented by the central nucleus. The basal and lateral nuclei have twice as many neurons per dimension as the central nucleus. Figure 3.13 shows the decoded basal nuclei values for models with 10, 100, 1,000, and 10,000 neurons per dimension. A model with 10 neurons per represented dimension has 1,360 total neurons in the model and 420 neurons in the amygdala. A model with 100 neurons per represented dimension has 4,600 total neurons in the model and 2,400 neurons in the amygdala. A model with 1,000 neurons per represented dimension has 37,000 total neurons in the model and 22,200 neurons in the amygdala. A model with 10,000 neurons per dimension has 3,610,000 total neurons in the model and 220,200 neurons in the amygdala. As illustrated by this figure, using too few neurons per dimension does not result in an accurate functional representation of the decoded value.

3.3.6.3 Biological Plausibility

To make the models more biologically plausible, a constraint was implemented such that each neuron in the model would only connect to other neurons with either all positive weights or all negative weights. Using Nengo, the least-squares minimization that calculates the appropriate connection weights to implement functions on the values represented by neural populations returns unconstrained connection weight values. In practice, the weight values tend to be normally distributed with a standard

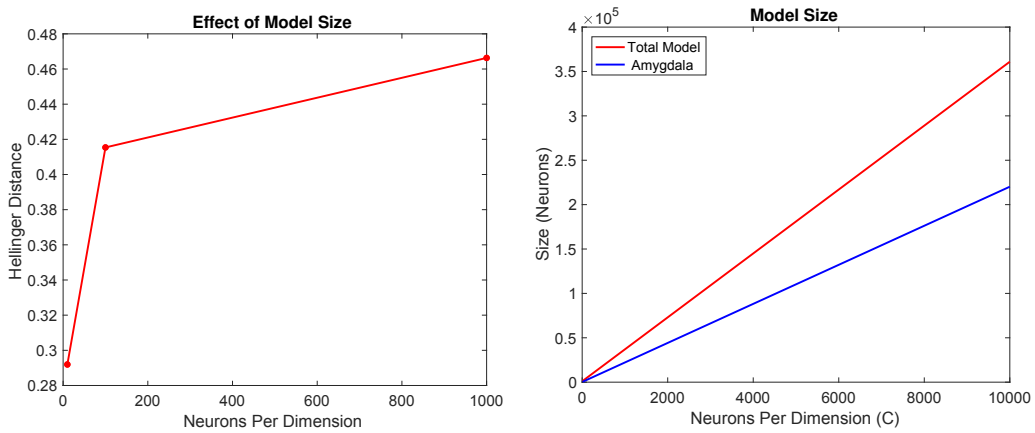


Figure 3.12: (a) Illustrates relationship between model size and the Hellinger distance to the primate amygdala neurons' response distribution. (b) Illustrates how the model size grows as the neurons per dimension (in the central nucleus) are increased. The model contains twice as many neurons per dimension in the basal and lateral nuclei, as in the central nucleus.

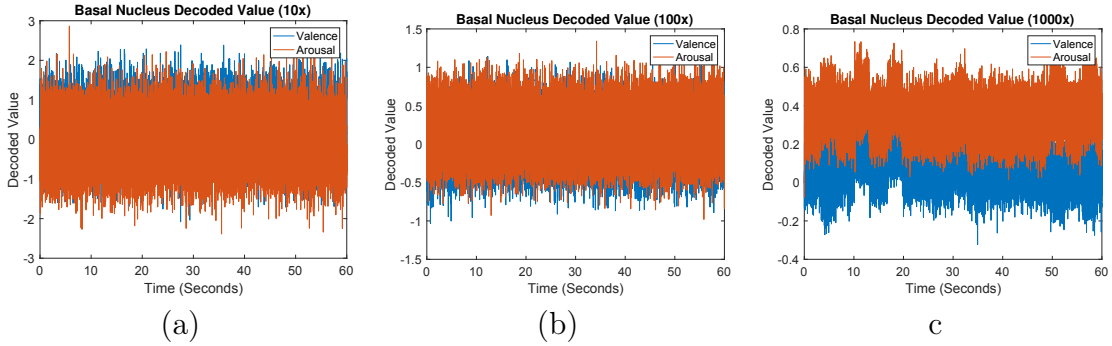


Figure 3.13: (a) Decoded representation of the basal nucleus when each dimension is represented with 20 neurons. (b) Decoded representation of the basal nucleus when each dimension is represented with 200 neurons. (c) Decoded representation of the basal nucleus when each dimension is represented with 2,000 neurons.

deviation that depends on the number of presynaptic neurons. Nonetheless, biological neurons are either excitatory or inhibitory and would never have some positively weighted (excitatory) and some negatively weighted (inhibitory) connections, which is why the model is constrained to avoid this. Additional constraints were also set

such that 80% of those connections were made using only positive weights and 20% of those connections were made using only negative weights to mimic the proportion of excitatory and inhibitory connections in biological neural systems.

To perform this optimization initially the non-negative least-squares (NNLS) optimization found via `https://github.com/alexfields/nnls` was utilized. This code could not be parallelized across multiple cores so instead an implementation of the non-negative least-squares optimization that supported parallelization was incorporated into the model code. This code can be found at `https://github.com/alexfields/nnls`. For a model of 37,000 neurons, where 12,200 were in the amygdala (1,000 neurons per central dimension), running the NNLS code took $\sim$ 15 hours to build the model. The NNLS code provided a tolerance parameter which could be varied depending on the accuracy of a solution. Because this calculation was costly in time, different tolerances were explored such that the model did not need to be built to an unnecessarily low tolerance which would result in prohibitively long build times. Figure 3.14 shows the results of this exploration and how different tolerance values affect the decoded represented values.

3.3.6.4 Sparsity

It is known that the different nuclei of the amygdala connect with different levels of sparsity, thus sparsity was also a parameter that was explored. By varying the level of sparsity, it was again explored how many connections were needed to accurately

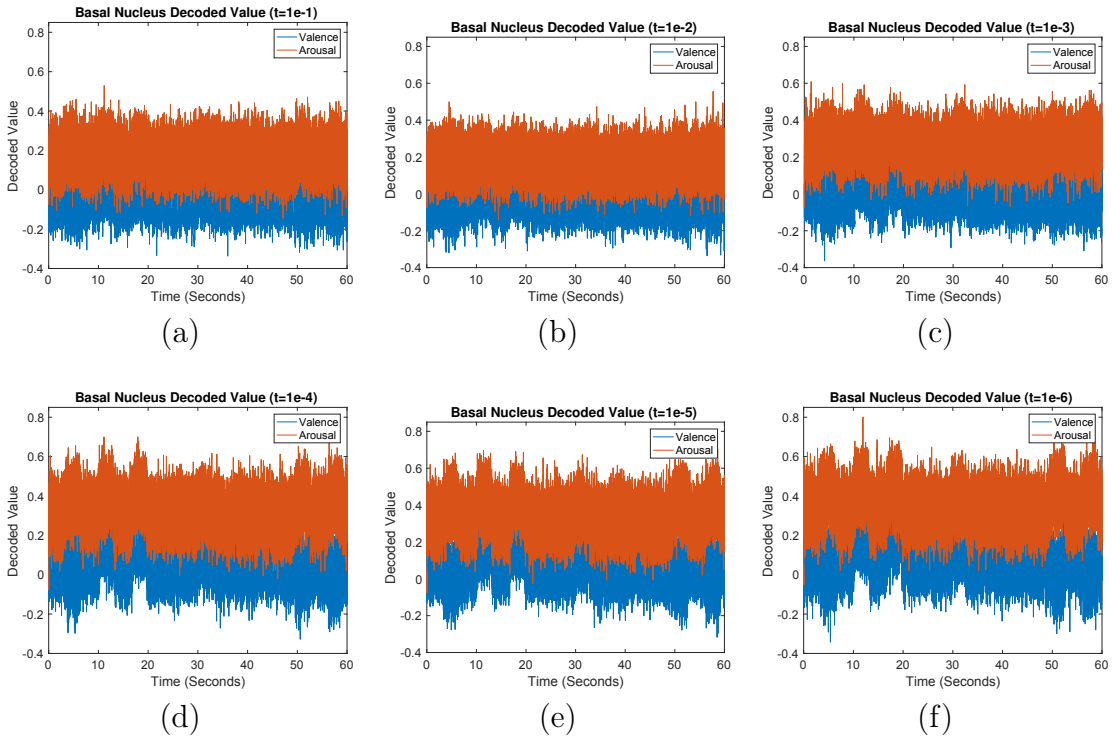


Figure 3.14: Plots illustrate the decoded value of the basal nucleus, when using different tolerance values for the non-negative least-squares (NNLS) optimization for calculating the model weights to provide greater biological plausibility. (a) Tolerance of 0.1 (b) Tolerance of 0.01 (c) Tolerance of 0.001 (d) Tolerance of 0.0001 (e) Tolerance of 0.00001 (f) Tolerance of 0.000001

represent values within the amygdala model, as well as how varying the amount of sparsity affected the Hellinger distance. Additionally, it was explored how varying the sparsity affected the build time of the model.

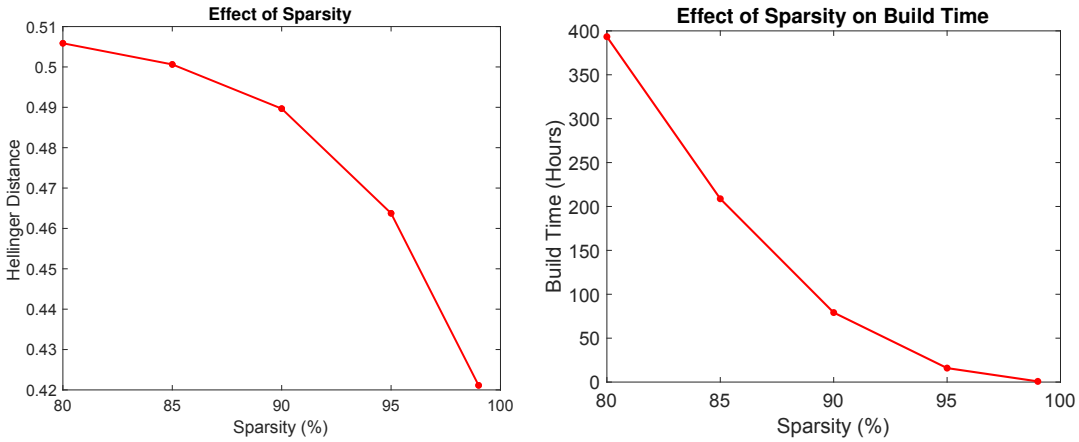
Figure 3.15 shows both the calculated Hellinger distance and the associated model build time for five models with differing levels of sparsity (99%, 95%, 90%, 85%, and 80%) but no other differing parameters. When it is said that a model has 90% sparsity, it indicates that for each post synaptic neuron in the model, 90% of its

incoming connections are deleted. In this plot the Hellinger distance decreases as the sparsity increases. We believe this is due to over-fitting of the model, but requires further explanation. Because of the additional constraint regarding excitatory and inhibitory connections, build times increase if the model contains more connections. The build time for the models with 99%, 95%, 90%, 85%, and 80% sparsity took 0.8, 15.9, 79.1, 208.8, and 393.3 hours, respectively. These models took 7.6, 7.6, 7.6, 7.7, and 7.7 hours to simulate, respectively. Build time length was one of the major limitations throughout the analysis of parameters.

Figure 3.16 illustrates the decoded two-dimensional basal nuclei value during a 60 second simulation for each of the models with 99%, 95%, 90%, 85% and 80% sparsity. The decoded value for the model with 99% is quite noisy, whereas the models with 95% and lower sparsity have visually distinguishable steps for the different states the basal nuclei represented. As expected, as more connections are utilized, the represented value is better approximated. Because a model with 95% sparsity had visually distinguishable states within the decoded value and it took significantly less time to build, this sparsity level was chosen as the one to hold while varying other model parameters.

3.3.7 Computational Considerations

As previously stated there are two main aspects of running a Nengo model: build and simulation. Initially both the build process and simulation process occurred on



(a) (b)

Figure 3.15: Effect of Sparsity on Hellinger distance and build time. (a) Hellinger distance for models with 99%, 95%, 90%, 85%, and 80% sparsity. (Tolerance = 1e-6). (b) Build time for models with different amounts of sparsity. The model with 99%, 95%, 90%, 85%, and 80% sparsity took 7.6, 7.6, 7.6, 7.7, and 7.7 hours to simulate, respectively.

the CPU of a 2015 Mac Book Pro with a 2.8 GHz Intel Core i7 processor at 2.8 GHz with 16 GB of memory. Once the model increased in size such that it exceeded the available memory of that laptop when simulating, models were then built and simulated on a Linux desktop running Ubuntu with 64 GB of RAM and an Intel Core i7-5960X processor at 3.00 GHz. To further speed up the build execution, models were later executed on a NVIDIA GeForce GTX TITAN Z graphical processing unit (GPU) with the help of the Nengo GPU backend, `https://github.com/nengo/nengo-ocl`.

Figure 3.17 shows how the Hellinger distance changed as a result of the number of cycle runs. Each cycle was $\sim$ 7 seconds of simulated experiment time. Each cycle included the time before the cue was shown, the cue visualization, the image visualization, and the inter-trial time.

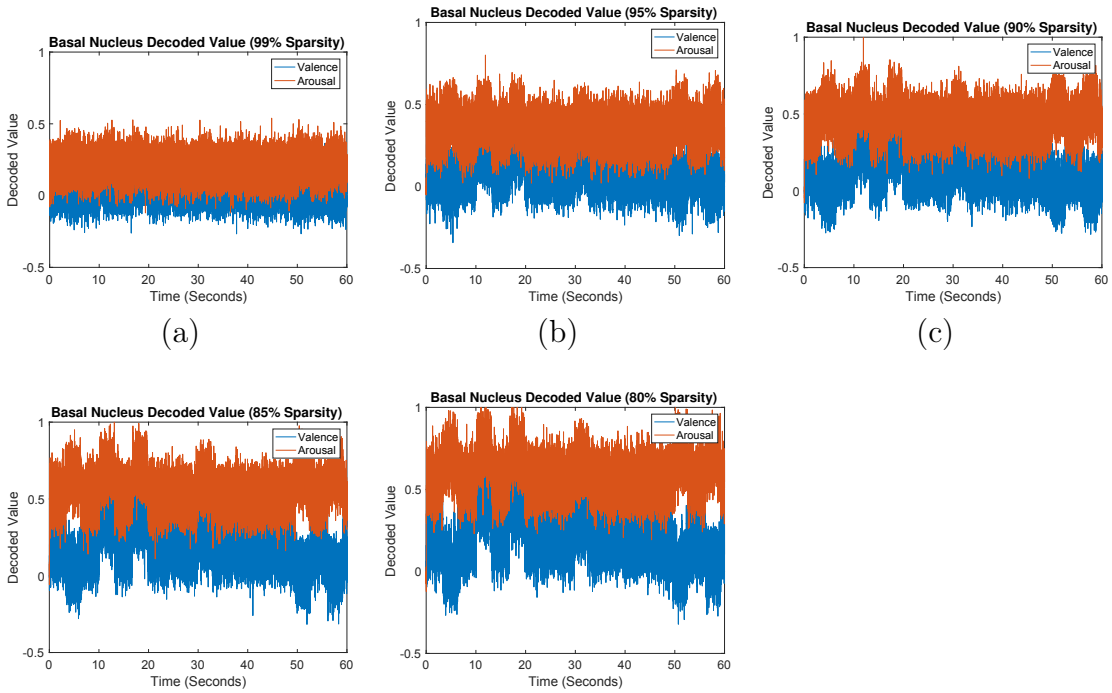


Figure 3.16: Effect of Sparsity on Decoded Representation. (a) 99% sparsity. (b) 95% sparsity. (c) 99% sparsity. (b) 85% sparsity. (b) 80% sparsity. 90% sparsity means that 90% of each post-synaptic neuron's connections are randomly deleted, and the connection is optimized with the remaining 10% of the connections. Plots only illustrate the decoded basal nucleus value for 60 seconds of simulation.

3.3.8 Model-Experiment Feedback Loop

The collaboration behind the work described in this chapter presents a unique opportunity to use this model as a tool to inform future primate neuroscience physiology experiments, which can subsequently inform future models. Already this work has prompted a number of conversations about interpretations and implications of model results.

One such example is illustrated within the contents of Figure 3.20. Figure 3.20 shows six neuron response plots. Three of these neurons were measured in the primate

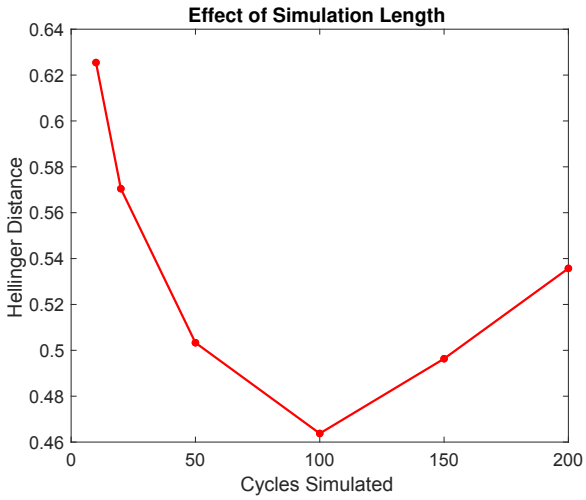


Figure 3.17: Illustrates the effect of model simulation time on the calculated Hellinger distance. Here each cycle is ~ 7 seconds of simulation.

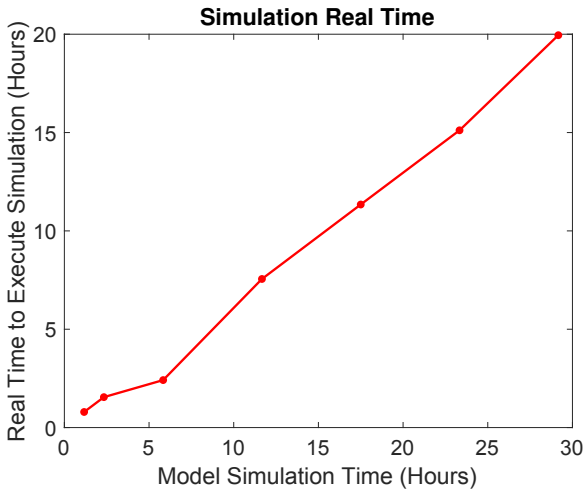


Figure 3.18: Illustrates the increase in time to simulate as the length of the experiment grows. For this plot experiments of 10, 20, 50, 100, 150, 200 and 250 cycles were simulated using a GPU. A cycle corresponds to about seven seconds (depending on the cue fixation time, which is drawn from a normal distribution since the primates did not always begin fixation at the same time). The number of cycles simulated was converted into simulation hours. These models took 15.0, 15.2, 15.2, 15.9, 14.8, 15.0, and 16.5 hours to build the 10, 20, 50, 100, 150, 200, and 250 cycle models, respectively. These models only deviated by chance and were built with the same parameters.

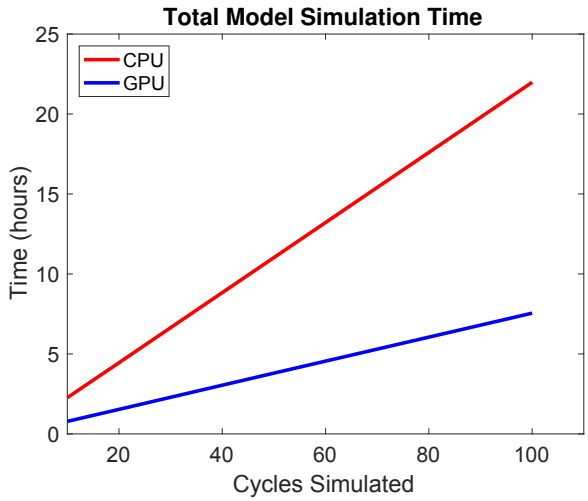


Figure 3.19: Illustrates the difference in execution time for the same model when using a CPU as compared to a GPU.

amygdala and three come from a model amygdala. The primate amygdala neurons all exhibit obviously phasic excitatory responses to the onset of the image. The three neuron plots that come from the model are characterized by the classification algorithm as exhibiting a phasic response to the onset of the image but visually do not appear to exhibit this response. After visually inspecting many of the model neuron response plots, it was concluded that the model does not produce visually obvious phasic responses to the image onset, as illustrated by the primate amygdala model. This conclusion led to a parameter exploration to potentially bring about these types of responses. The inhibitory and excitatory time constants were varied to attempt to bring about phasic responses. Phasic responses occur because of the time differential between the excitatory and inhibitory responses. Inhibition generally follows a longer time constant, kicking in after the excitatory response, which is what produces the phasic spike in firing rate. Varying the excitatory and inhibitory time constants did

produce phasic responses in the model. Moreover, this begs a longer conversation about why these responses are measured in the amygdala. Does the model lack some functionality that produces these responses? Do these responses originate in the amygdala? Some of these questions can be further probed by the model, whereas others may benefit from future primate neurophysiology experimentation.

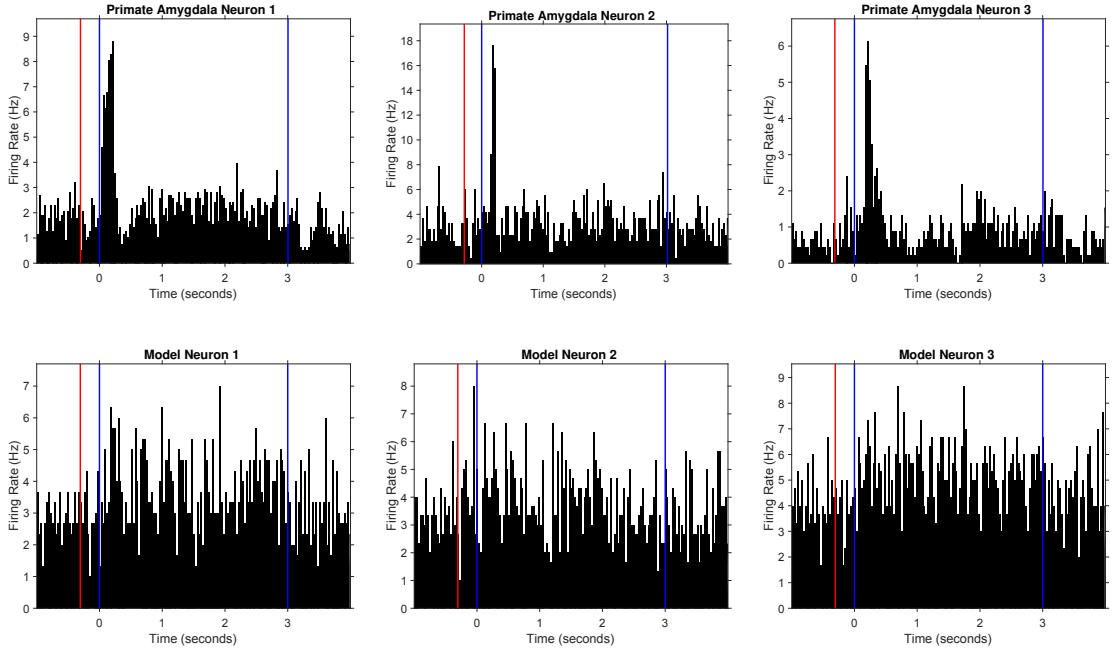


Figure 3.20: The top row shows three primate amygdala neurons. All three neurons exhibit a visually obvious phasic excitatory response to the onset of the cue. The bottom row shows three neurons from the model that are classified as having a phasic excitatory responses to the cue. Besides the fact that the classification algorithm could always be improved, the lack of any phasic excitatory responses to the experiment brings a larger question to light. Why does the model lack these responses? This is one of the many questions the model poses which may benefit from future primate neurophysiology experiments.

3.4 Methodology Challenges

In addition to computational challenges that arose as the model increased in terms of the number of connections and number of neurons, running a Nengo simulation to replicate a primate experiment presented some challenges to the Nengo infrastructure, which were overcome to enable this work. This work presents the first model built using Nengo and then validated with real neural data. In Nengo when one probes or records a neuron's spike times, the software creates an array, s, of length t where t is the number of simulation time steps. $s_i(t) = 1$ when there was a spike emitted by the ith neuron at time, t. Otherwise $s_i(t) = 0$. Initially the model would fail for any simulation with over 3,000 neurons. Saving spike responses as arrays of 1's and 0's is not an efficient way to store spike responses. Instead a spike node was created and the time of each spike emitted by each neuron was instead saved for later analysis.

Chapter 4

Applications on the TrueNorth Neurosynaptic System

4.1 TrueNorth Architecture Overview

The IBM TrueNorth Neurosynaptic processor has a tightly coupled memory-processor architecture[13] *. It contains 4096 event-driven cores which operate using custom asynchronous and synchronous logic. They are globally connected using an asynchronous packet switched mesh network on chip (NOC). TrueNorth sits on a development board which includes a Xilinx Zynq Z-7020 SoC (system on chip) that provides communication support and housekeeping. The NS1e development board is shown in Figure 4.1.

*Sections from this chapter were previously published by the author.[146–148]

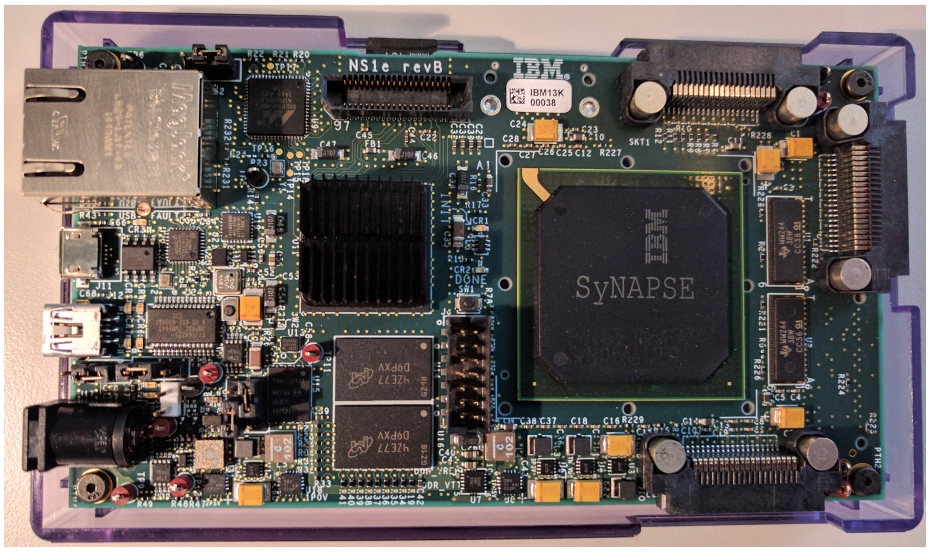
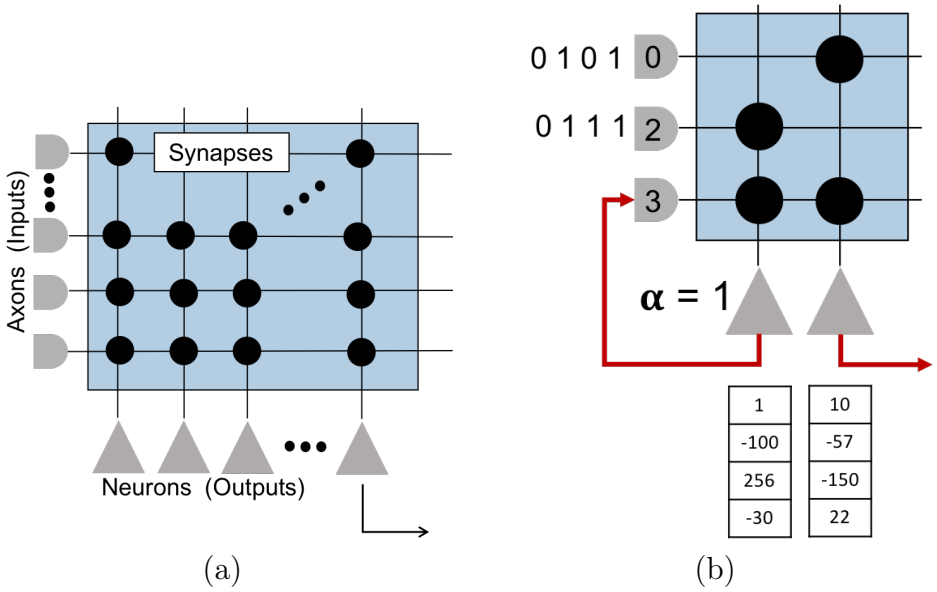


Figure 4.1: The NS1e evaluation platform which houses the TrueNorth chip. The NS1e has a Xilinx Zynq Z-7020 providing two ARM Cortex-A9 cores and configurable FPGA fabric. It is 125 mm x 690 mm and weighs 98 g.

Each TrueNorth core contains 256 inputs (axons) and 256 outputs (neurons) connected using a configurable crossbar array as seen in Figure 4.2a. Each connection, denoted by a binary synapse value, is assigned a synaptic weight, found in a lookup table associated with the neuron to which it connects. Each neuron, or column of the crossbar array, has a unique lookup table with four integer value synaptic weights in the range [-255, 255] and multiple programmable parameters including membrane potential, leak, spiking threshold, destination axon, and delay. These parameters and the crossbar array configuration are declared using IBM's hierarchical, compositional programming language, Corelet Programming Environment (CPE).[31] Computational units are defined and connected using CPE and can then be executed directly on TrueNorth or in Compass, a scalable simulator of the hardware.[149] Parameters can-



(a) (b)

Figure 4.2: Illustration of a basic TrueNorth core adapted from the author's previous work.[147] TrueNorth cores have 256 inputs (axons) and 256 outputs (neurons), although (b) shows only 3 axons and 2 neurons. In (b) at time $t = 1$ spikes flow into the crossbar from both the first and second axons. Where there is a synaptic connection in the crossbar, each spike is multiplied by the corresponding weight found in the look-up table under the corresponding neuron column. Because the first axon is Type 0, the synaptic connection between it and the second neuron has a weight of 10. Because the second axon is of Type 2, the connection between it and the first neuron will have a synaptic weight of 256. In this diagram the neurons have a threshold, α, of 1 which means that each neuron will emit a spike once the membrane potential reaches a value of 1. Following the reset rule used in this example, after emitting a spike, the neuron membrane potential will decrease by the threshold, or 1. The first neuron's spikes are sent to the third axon of this crossbar, whereas the second neuron's spikes travel to an axon on a different crossbar.

not be dynamically updated and are therefore only programmed prior to run-time.

Figure 4.2b shows a basic TrueNorth core with only 3 axons and 2 neurons. To the left of each axon, spikes are depicted. On TrueNorth, each axon has a spike buffer which is used to delay spikes up to 15 time steps from the time a source neuron releases them to a destination axon. In Figure 4.2b, at time $t = 1$ spikes flow into the

crossbar from both the first and second axons. Where there is a synaptic connection, the spikes are multiplied by the corresponding weights found in the look-up table associated with each neuron column. Because the first axon is Type 0, the synaptic connection between it and the second neuron has a weight of 10 ($0th$ element in the look-up table). Because the second axon is of Type 2, the connection between it and the first neuron will have a synaptic weight of 256 ($2nd$ element in the look-up table). In this diagram the neurons have a threshold, α, of 1 meaning that each neuron will emit a spike once the membrane potential reaches a value of 1. Following the reset rule used in this example, after emitting a spike the neuron membrane potential will decrease by the threshold, 1. The first neuron's spikes are sent to the third axon of this crossbar, whereas the second neuron's spikes travel to an axon on a different crossbar. Spikes are released through the axons into the crossbar arrays according to a global synchronization clock, which counts "tick" time steps. A tick is roughly equivalent to one millisecond and corresponds to the global time signal by which spikes reach their destination. Membrane potentials are aggregated each time step.

4.2 TrueNorth Use Cases

There are two main ways that TrueNorth can be programmed and used, one which focuses on implementations of convolutional neural networks, and another which focuses on implementations of custom algorithms. To allow TrueNorth hardware to eas-

ily execute simulations of convolutional neural networks, the TrueNorth development team created a deep learning toolkit called Eedn.[30] Eedn stands for "Energy-efficient deep neuromorphic networks" and is a framework for training convolutional neural networks constrained to run on TrueNorth's underlying architecture. Using Eedn, users specify the layers of their convolution neural network in MATLAB. Then Eedn trains the network using a constrain-then-train approach, constraining the network to TrueNorth's specific hardware. After training, the trained network can be loaded onto TrueNorth to execute. Eedn trains networks using MATLAB and the MatConvNet deep learning framework. Work by this author employing Eedn is described in Section 4.4.

The second way to use TrueNorth is to develop corelets using IBM's hierarchical, compositional programming language, Corelet Programming Environment (CPE).[31] Using the CPE, users can describe crossbar configurations, neuron parameters, core inputs, and core outputs to produce a specific computation through the processing of spikes. Users can link multiple cores to perform these computations in succession, essentially developing spike-based algorithms for TrueNorth. Programmed corelets can be executed directly on TrueNorth or in Compass, a scalable simulator of the hardware.[149] Custom corelet design by the author is described in Section 4.3 and in Chapter 5 Section 5.4.1.

4.3 Path Planning on TrueNorth

IBM's Neurosynaptic System, also known as the TrueNorth chip,[13] offers the opportunity to explore bio-inspired path planning algorithms on an industry supported, general purpose neuromorphic hardware platform[†]. This platform's low power consumption and programmable, spiking neurons offer an ideal hardware system on which to implement robotic navigation applications. Prior work of path planning algorithms on neuromorphic hardware use spiking VLSI neurons,[150,151] where an array which fits a map of 4 to 100 neurons or nodes is used to propagate spikes throughout the map. An associated microcontroller stores the address event representation (AER) timing information of the silicon neurons' spike activity. The optimal path is later determined using this stored timing information. This section describes the implementation of a wave-front algorithm for path planning[152] on the TrueNorth chip[13] that employs a neuronal spike-wave to traverse a map, stores timing information, and finds an optimal path. This work expands from previous implementations by performing all aspects of path planning on-chip, including storing the AER information and deducing the optimal path using spiking neurons. An illustration of the implemented wave front algorithm[152] can be seen in Figure 4.3.

[†]The contents of this section were previously published by the author.[148]

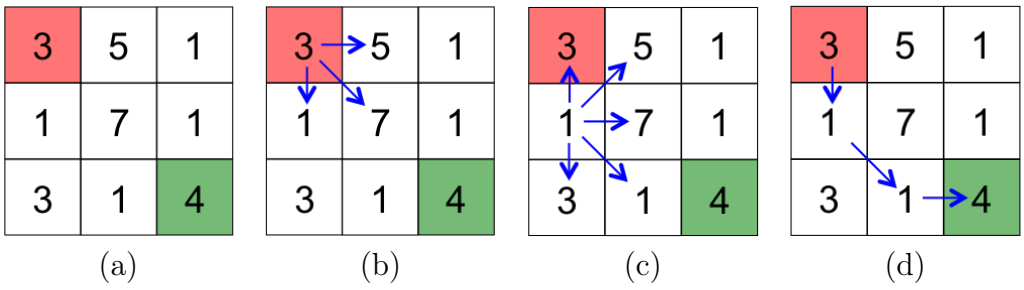


Figure 4.3: Spike-wave propagation using Krichmar's algorithm.[152] (a) Each node's cost is indicated. Start and end nodes are colored red and green respectively. For grid maps, neighbors are geographically co-located and middle nodes, edge nodes, and corner nodes have 8, 5, and 3 neighbors respectively. (b) First spikes at t=3 after a delay equal to start node's cost. (c) Spike propagation at t=4. Spikes have not yet propagated out of the nodes with higher costs. (d) The optimal path found by analyzing AER information.

4.3.1 Algorithm and Implementation

The implementation consists of four steps: (1) spike-wave propagation, (2) tie-break, (3) on-chip AER, and (4) path-finding. Each map location, or node, is assigned a dedicated portion of a TrueNorth core's crossbar array to implement these functions in parallel. A crossbar array consists of inputs (axons) connected to outputs (neurons) with a number of programmable parameters including synaptic weights, membrane potentials, spiking thresholds, and delays. TrueNorth neuron parameters were chosen to deduce and display the optimal path as the output of the system.

4.3.1.1 Spike-Wave Propagation

Spike-wave propagation is well-suited for implementation on the TrueNorth architecture, as TrueNorth neurons have the ability to encode delays with outgoing spikes

to represent the cost of traversal of a map node. Within each node's portion of the crossbar array, axons and neurons correspond to incoming and outgoing neighbors of the node respectively. An example of the implemented spike-wave propagation on TrueNorth is shown in Figure 4.4. When a neuron emits a spike from a node's section of the crossbar, this means that the spike-wave has reached this node. At this point, an axonal delay is encoded with the outgoing spikes by placing them in the destination axon's buffer location equivalent to the cost of traversing the node before proceeding to its outgoing neighbors. TrueNorth axons have buffers for delays up to 15 "ticks". Each "tick" is roughly equal to one millisecond. Thus, maps are quantized between 1 and 15 to fit onto one buffer. More resources can be used to create a series of buffers to increase the delay.

4.3.1.2 Tie-Break

There may be more than one route with the same cost to reach the end node. Synaptic connections and weights are leveraged to emulate a tie-break function for each node. If a node receives spikes from incoming neighbors simultaneously, only one is selected as the "winner". In Figure 4.5a, if the node received spikes from incoming neighbors A and C at the same time, only the first neuron representing neighbor A will spike by reaching its threshold of 1, whereas the second neuron representing C will be inhibited by a synaptic weight of -10. The output spikes are fed back to disable the neurons from processing any other inputs using synaptic connections with weights

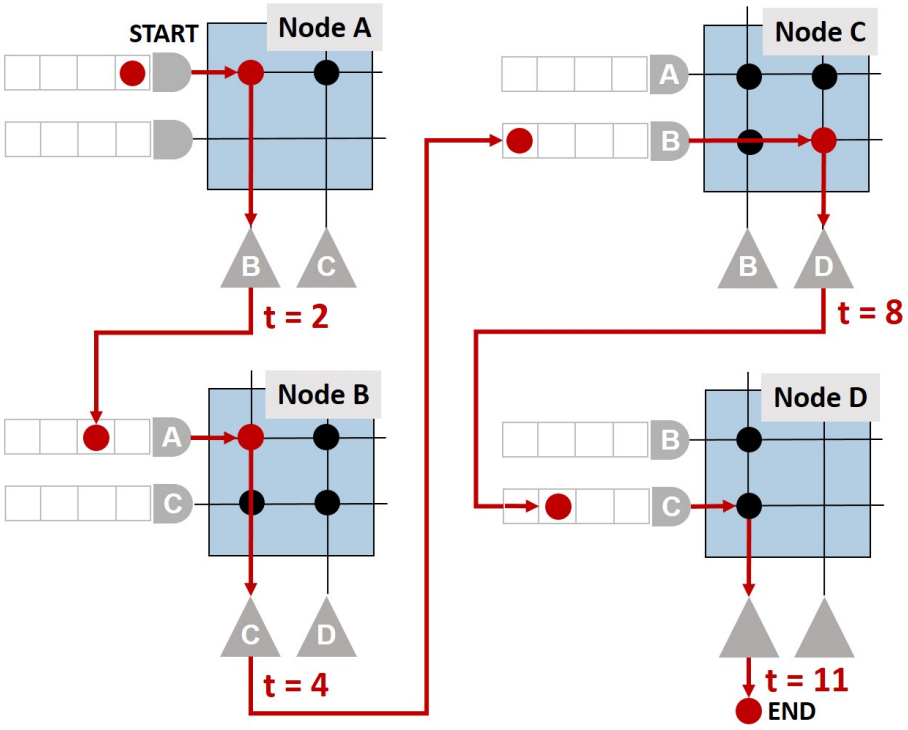
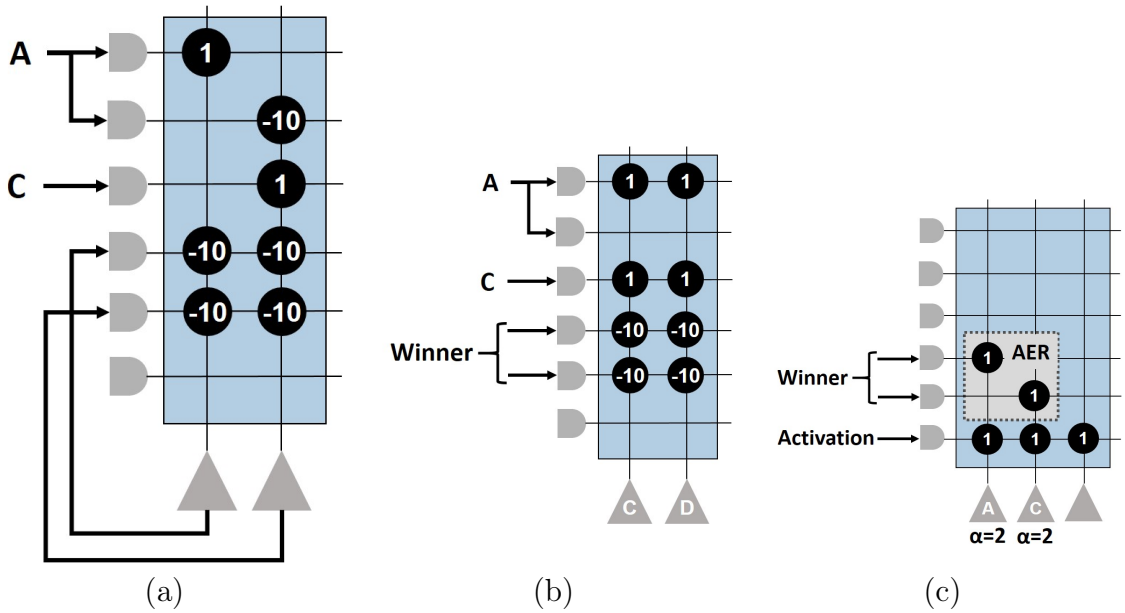


Figure 4.4: Spike-wave propagation of a 2×2 map using 4 designated portions of a core's crossbar. Neurons are indicated with triangles, axons with half-circles. Cost of nodes A, B, and C are 2, 4, and 3 respectively. Outgoing neighbors of nodes A, B, and C are {B,C}, {C,D}, and {B,D} respectively. Simulation begins at t=1. Spike propagation is shown only for path A → B → C → D for clarity, whereas in this example spikes would emit from every neuron in parallel.

of −10, as only the information about which incoming neighbors' spikes reached each node first needs to be tracked. The spike-wave propagation then proceeds with only one stored incoming neighboring node as the "winner". Architectural constraints of the TrueNorth chip limit each neuron output to one destination axon input, hence neurons must be replicated to perform the spike-wave propagation tie-breaking process for the proceeding nodes.



(a) (b) (c)

Figure 4.5: A node's designated portion of a TrueNorth crossbar with incoming neighbors {A,C} and outgoing neighbors {C,D}. Synaptic weights are overlaid on each connection. All thresholds are α=1 unless otherwise noted. The crossbar is separated for visual clarity. (a) Tie-break neurons. (b) Spike-wave propagation neurons, same as processed in Figure 4.4. Feedback from tie-break neurons is used to inhibit spike-wave propagation neurons to emulate a neuronal refractory period. (c) On-chip AER and path-finding neurons.

4.3.1.3 On-Chip AER

Neuron potentials and thresholds are used to store an AER table on TrueNorth. Once the tie-break function is implemented for a node, the "winner" is fed back to the node's crossbar, shown in Figure 4.5c. The first two neurons represent incoming neighbors A and C and have thresholds $\alpha = 2$. If incoming neighbor A is declared the winner, the neuron representing A will remain at a potential of $V = 1$ until the node is activated as part of the optimal path. The neuron representing neighbor B will have a potential $V = 0$ and will therefore never spike even if the node is activated

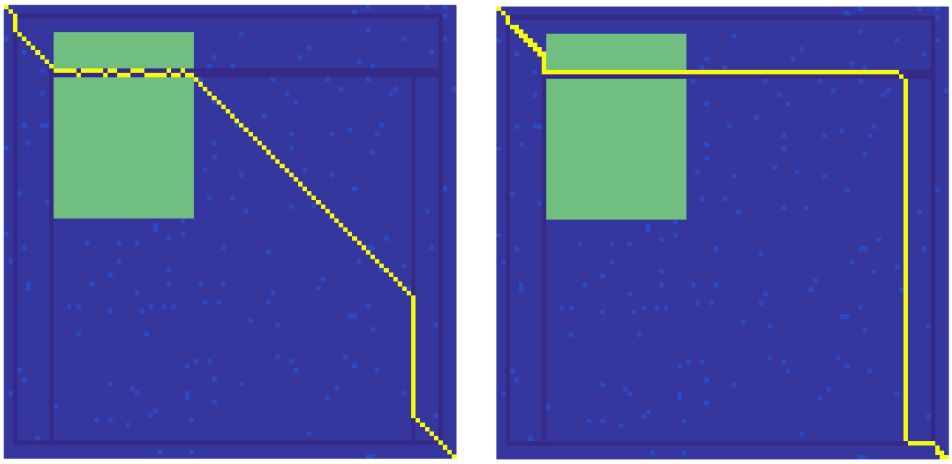
since we use inhibition to guarantee only the first spike into a node is processed.

4.3.1.4 Path-finding

Once the propagating spike-wave reaches the end node, all nodes have one winner stored. Each node that lies in the optimal path will receive a spike via its activation axon, the last axon in Figure 4.5c. This increases the neuron's potential from $V = 1$ to $V = 2$. A spike is then sent from that neuron to the activation axon of the winning incoming neighbor's crossbar. This process begins at the end node and continues in reverse order along the optimal path until the start node is reached. The last neuron in Figure 4.5c is also connected to the activation axon. This neuron sends an external output spike if this node is part of the optimal path. The external output spikes of all nodes found to be in the optimal path illuminate the ideal path to traverse the map.

4.3.2 Experimental Results

For any random map, our path planning implementation on the TrueNorth chip calculates a path with a cost less than or equal to the results from Krichmar's algorithm[152] as we would expect. Differences arise due to randomness associated with tie-breaking. We find the optimal path by displaying the output spikes from the TrueNorth chip and show that these results compared to the output of Krichmar's algorithm[152] in Figure 4.6.



(a) (b)

Figure 4.6: (a) The optimal path (in yellow) of a map as calculated by the TrueNorth chip with a total cost of 196. (b) The optimal path as calculated by Krichmar[152] with a total cost of 216.

This implementation can find the optimal route for the largest maps to date on neuromorphic hardware. The designated portion for each map node on the crossbar array is a function of incoming neighbors N_{in} and outgoing neighbors N_{out}, resulting in $(N_{in}*3)$ axons and $(N_{in}*2+N_{out}*2+1)$ neurons per node. Multiple nodes' crossbar portions can be tiled onto a single core to maximize implementation efficiency and result in solutions for larger maps. The resources required to analyze maps of different sizes are visualized in Figure 4.7. For a single TrueNorth chip spikes are propagated, an AER is stored, and the optimal path is deduced for grid maps with dimensions up to 173×168 nodes. If larger maps are needed, multiple TrueNorth chips can be connected in series allowing for a proportional increase in map size and no changes to the algorithm. If only spikes are propagated on the TrueNorth chip and the AER data is saved to analyze off-chip after, this method analyzed grid maps with dimensions

up to 338×340 nodes using one TrueNorth chip.

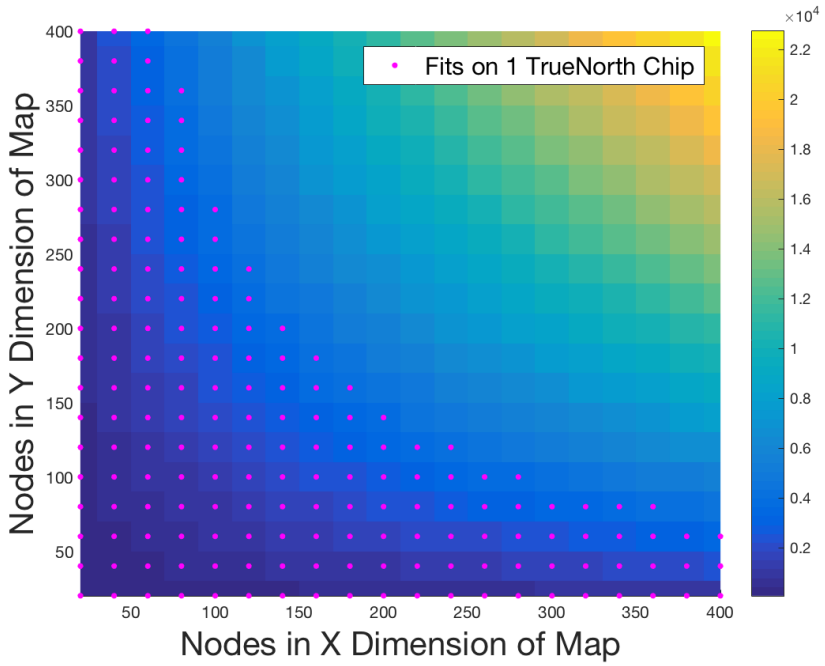


Figure 4.7: Number of nodes in each dimension of the grid map versus TrueNorth cores required to implement. Maps that can be contained on a single chip are denoted with a magenta dot.

4.3.3 Computational Time and Power

Spike-wave propagation requires a number of ticks equal to the total cost of the optimal path. Deducing the optimal path requires ticks equal to number of nodes in optimal path. The largest maps this mapping allows on the TrueNorth chip with average costs of 4.34 ticks per node require an average run-time of 577.3 ticks. Maps with 10×10 nodes at an average cost of 3.65 ticks per node have an average run-time of 37.2 ticks. Each tick is roughly equal to one millisecond.

Four grid maps with dimensions of 25×25 nodes, 75×75 nodes, 125×125 nodes, and 173×168 nodes utilize 2.0%, 19.0%, 53.5%, and 100% of the cores available on one TrueNorth chip respectively. These maps respectively utilize an average of 1.4mW, 13.2mW, 27.2mW, and 70.0mW at an operating voltage of 0.8V and 2.9mW, 28.2mW, 77.9mW, and 144.5mW at an operating voltage of 1V. The total power is calculated by scaling the leakage power by the number of cores actually used, where P is power and $P_{total} = P_{active} + P_{leak} * N_{cores}/4096$.[153]

4.3.4 Extensions to Topological Maps

This implementation offers the flexibility to extend analysis to topological maps. Each node can have any number of incoming and outgoing neighbors independent of node location on the map. The incoming neighbors can be different nodes than the outgoing neighbors and all costs represent the relationship between only those two nodes. If it is assumed that there are eight incoming and eight outgoing neighbors for each node, a topological map with 28,674 nodes can be analyzed on one TrueNorth chip. For maps with nodes connecting to two incoming and two outgoing neighbors, the optimal path can be computed for maps up to 114,688 nodes.

4.3.5 Conclusion

This section demonstrates implementation of Krichmar's path planning algorithm[152] on the TrueNorth neurosynaptic system. This approach not only propagates the spike-wave through the map, but also collects timing information and deduces the optimal path on-chip. The system consumes ∼70 mW at an operating voltage of 0.8 V for maps with dimensions up to 173×168 nodes, offering a realizable opportunity for path planning applications on embedded systems and autonomous robotic applications.

4.4 Neuromorphic Self-Driving Robot Using TrueNorth

Neuromorphic hardware departs from the sequential processing of Von Neumann architectures by using a massively parallel design to mimic biology, reducing processing time and overall power consumption[154–156‡]. Neuromorphic platforms like IBM's TrueNorth Neurosynaptic System[13,153] provide portable neuromorphic architectures perfect for use in autonomous system applications or other real-time, power constrained environments. The TrueNorth chip is a great platform choice for spike-based machine learning implementations due to its spike-based computational abilities and brain-inspired, low-power hardware design. Deep learning with convolutional neural networks (CNNs) is particularly suitable to this platform because of IBM's energy-

[‡]The content of this section was previously published by the author.[147]

efficient deep neuromorphic networks (Eedn) software framework for training the hardware to efficiently run these networks with the hardware constraints posed by the chip.[30]

This section details an extension of the self-driving robot described by Hwu.[157] However, this implementation departs from previous work because it uses an Asynchronous Time-based Image Sensor (ATIS)[158] to provide input to TrueNorth instead of an Android smart phone. This spike-based sensor is capable of communicating events from the visual scene in the native processing domain of TrueNorth, resulting in the first fully neuromorphic self-driving platform.

4.4.1 Background

Machine learning is a growing field used in a wide range of applications:[159–162] from customizing web browsers by learning user interests to automatic speech recognition, stock market prediction, image classification, and more recently autonomous driving vehicles. In some machine learning techniques, human extracted features are processed with machine learning algorithms. The endeavor of feature extraction requires a great deal of time and effort on behalf of trained engineers. This tedious and time-consuming exercise inherits the "human bias" which may result in neglecting important features which could be critical to system performance. This effect may be mitigated using deep learning, in which new data representations are automatically learned, thus enabling computer algorithms to interact with raw data while

maintaining system performance.

This work applies deep learning techniques to the problem of autonomous navigation. This emerging field encompasses image processing, machine learning, and robotics, implementing a closed loop system whose decisions are based primarily on raw input images. Pomerleau introduced an autonomous land vehicle in 1989[163] that utilizes a 3-layer neural network, making it one of the first machine learning approaches in autonomous navigation. More than a decade and a half later, the Defense Advanced Research Agency (DARPA) introduced the DARPA Autonomous Vehicle (DAVE),[164] which demonstrated autonomous navigation in an alley of obstacles without communication to a home base. The system consisted of a 6-layer convolutional neural network (CNN) and was trained from end-to-end by mapping raw images–taken while the system was driven by a human–to steering angles. A more recent approach, DAVE-2, was proposed by one NVIDIA research group.[162] Interestingly, their vehicle was able to navigate areas under different weather conditions and with limited visibility. Their 9-layer CNN managed to detect features suitable for predicting vehicle movement.

The aforementioned projects raised awareness for future directions within this field since they dealt with multiple challenges such as changes in the environment, weather, and terrain, as well as obstacle avoidance. However, the CNNs were usually deployed on bulky GPUs consuming hundreds of watts and were powered by heavy batteries. As previously mentioned, a group at University of California Irvine (UCI) created a

low power approach to address this problem.[157] They used a deep CNN implemented on IBM's TrueNorth Neurosynaptic System. However, their data-processing pipeline used images captured from an Android smart phone, so consequently an extra layer of post-processing had to be implemented to map raw pixels to spikes for input to the TrueNorth chip. Our implementation instead uses the ATIS to produce spikes directly from the visual scene, resulting in a fully neuromorphic system assumed to consume less power. Details of this system are described in the next section.

4.4.2 Robot Platform Hardware

The robot platform (see Figure 4.8) was built using guidelines from the University of California, Irvine (www.socsci.uci.edu/∼jkrichma/ABR/) and consists of a six WD aluminum differential drive chassis propelled by six motors using 75:1 gearboxes. The chassis payload allows all data collection hardware to be easily mounted. The motors are controlled by a 2x30A controller which accepts drive and steer commands from an Android smart phone mounted on a pan and tilt system. This phone runs a host application, which in turn receives commands from a client tablet running a controller application. The former is adapted from *Android-Based Robotics* code (https://github.com/UCI-ABR) and provides discrete driving commands, i.e. discrete speed and discrete turning angles. This approach allows the robot to maneuver in tight spaces as well as open areas while providing clear, unambiguous driving commands (labels) for training a CNN. Finally, the host phone communicates with the

motor controller via a special development board, specifically a SparkFun IOIO microcontroller, designed to interface with Android platforms. Figure 4.8 shows the front of the chassis supporting a pan servo-motor, which holds the ATIS.

(a) (b)

Figure 4.8: Robot data collection platform: front (a) and side (b) views. Notice the ATIS mounted on the front, the smart phone in the middle, and the Surface Pro mounted on the motor controller box.

ATIS is a custom CMOS vision sensor with 304×240 pixel resolution. Each ATIS pixel operates independently, detecting changes in its own log-illumination, and outputting an event whenever such a change is detected. An event consists of the pixel address and 1 bit of polarity to indicate whether the change was an increase or decrease in log-illumination. These events are communicated off-chip with very low latency, and therefore the time at which an event is output by the ATIS can be considered as the time at which the corresponding change in log-illumination was detected.

The ATIS is controlled by a Xilinx Opal Kelly XEM6010-LX45 board, which allows for logging of event based data onto a Microsoft Surface Pro. The entire ATIS

system is powered by the main robot battery and consumes less than 5 W. During data collection, in addition to storing the ATIS data on the Surface Pro, the motor pulse-width modulation (PWM) control signals and photos taken from the Android smart phone of the scene are stored on the Android phone. Data are processed later to form the training input for the deep CNN.

4.4.3 Real-Time Autonomous Platform

The real-time system operates similarly but does not require the Surface Pro since the TrueNorth chip can ingest spikes directly and there is no need to log data. In this system, the ATIS spikes are sent to a portable 12 V-powered Linux PC mounted on the robot which is powered with another battery. This PC simply groups spikes together as described in Section 4.4.6 and sends batches directly to the TrueNorth hardware using TCP communication. The TrueNorth chip pushes the spikes through the trained network stored on board and then outputs the results back to the portable PC. Finally, the PC decodes these results and commands the motor controllers to move the wheels in the appropriate fashion. Figure 4.9 shows how information flows using this platform. The autonomous system bypasses the Android phone and tablet which were previously used to control the robot when gathering training data since these devices are not required when the robot drives by itself.

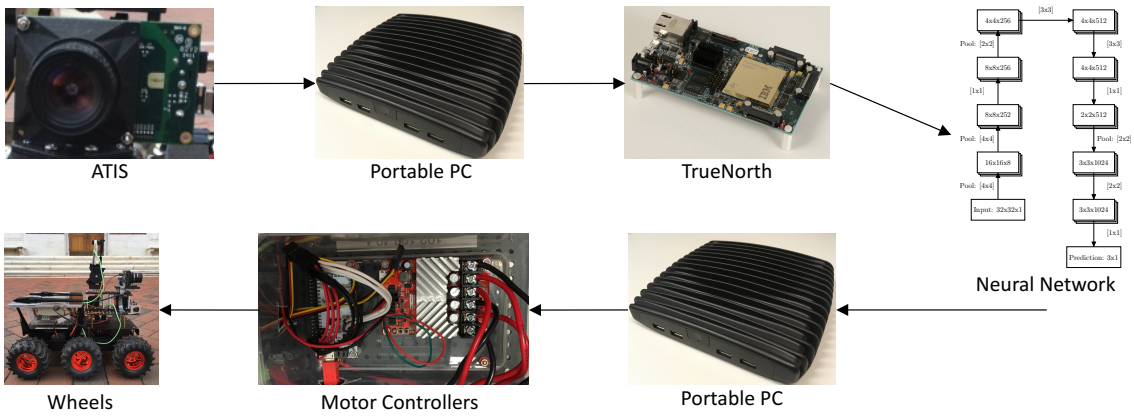


Figure 4.9: Autonomous platform data flow.

4.4.4 Data Collection

Data was collected from three different environments including the hallways of a building at Johns Hopkins University, campus sidewalk paths (approximately 2 meters wide), and residential sidewalks (approximately 0.6 meters wide). Figure 4.10 shows images from the Android smart phone camera taken during data collection as well as the processed ATIS output.

4.4.5 Data Pre-Processing

The main challenge in data pre-processing was time-domain alignment between asynchronous-time address events and uniformly sampled (usually 30 Hz) driving commands from the host smart phone. Due to the different nature of the recording systems, time stamps were aligned offline using a custom algorithm, explained below.

On the smart phone data side, data was truncated to the onset of movement which

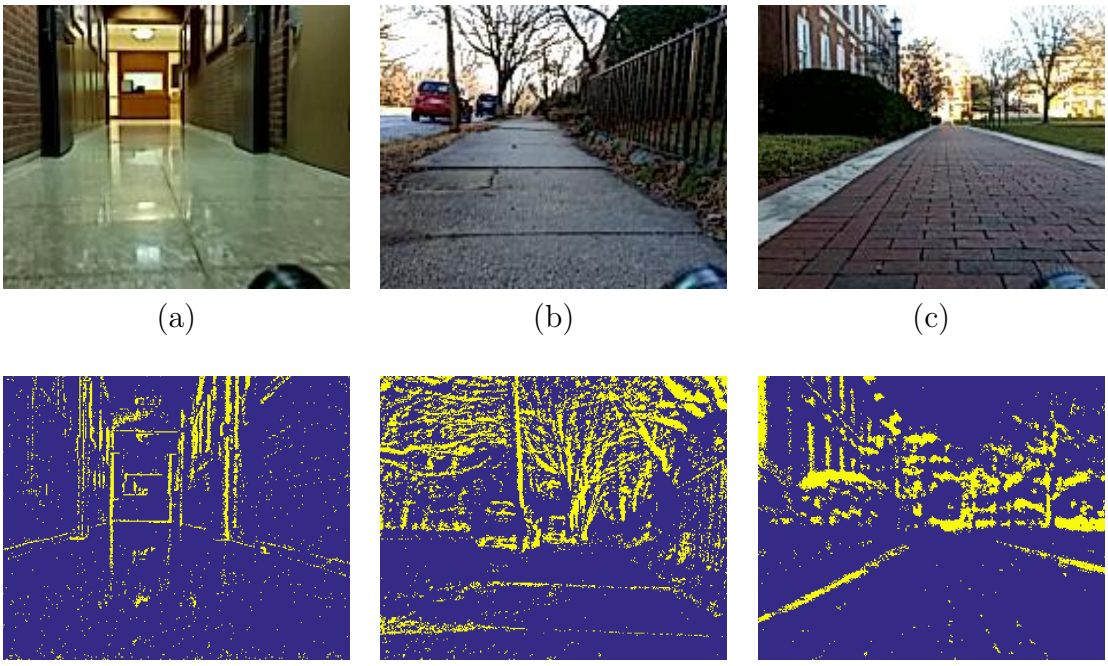


Figure 4.10: Examples of the three collected data sets. (a) Data collected using the Android phone while the robot was driven in the hallways of a building at Johns Hopkins University and (d) the corresponding input to Eedn. (b) Data collected using an Android smart phone while the robot was driven on a residential sidewalk and (e) the corresponding input to Eedn. (c) Data collected using the Android smart phone while the robot was driven on a campus sidewalk path and (f) the corresponding input to Eedn.

was found by identifying peaks in the first derivative of the robot speed, derived from raw PWM signals. On the ATIS side, the onset of movement was captured by searching peaks in the derivative of the inter-spike-intervals (ISI). When the robot was motionless, spikes were rare, mostly due to noise or changes in light. Conversely when the robot was moving, the time-domain differences in the scene relative to the robot created a high firing rate, thus a lower inter-spike-interval. As a result, the first major peak in the derivative of the inter-spike-interval revealed the onset of

movement. After the data alignment, spline interpolation was used to append labels to the AER data.

4.4.6 Network Training

After the ATIS output spikes were assigned labels, they were used as input to the training of a deep learning neural network trained to output the correct driving direction (left, right, center). Training was performed using IBM's energy-efficient deep neuromorphic networks (Eedn) software framework for TrueNorth.[30] Eedn trains convolutional networks, whose connections, neurons, and weights have been constrained to map onto the TrueNorth chip. Training produces a corelet, or composable hardware description, that can be programmed directly onto the TrueNorth chip.[31] Eedn allows the user to specify the parameters of each layer within the network, including number of features, kernel sizes, padding, stride size, learning rates, and many others. While training the network, Eedn can train and test either within Compass, a one-to-one scalable software simulator, or directly on the TrueNorth chip.[149]

Because the ATIS generates spikes in very fine-grained intervals, to train the network in Eedn, spikes are sent into the system in batches, similar to the way spikes are delivered during real-time communication. Spikes were aggregated within a specific time window to generate spike-frames. Each of these groups (frames) was assigned a label based on the majority of occurrences within that time frame. Once the spikes were grouped and labeled, they were divided into train, development, and

test sets. The training set consisted of roughly 60% of the collected data while the development set and test set were roughly 20% each. The hallway dataset contained 6,591 training groups, 2,242 development groups, and 2,189 testing groups. The campus sidewalk dataset contained 18,076 training groups, 6,099 development groups, and 6,071 testing groups. Finally, the residential sidewalk dataset contained 12,865 training groups, 4,347 development groups, and 4,271 testing groups.

4.4.7 Results

4.4.7.1 Preliminary Results

Preliminary work included training a CNN on GPUs using the MATLAB Neural Network Toolbox. Two different datasets were used to train the same network. The network architecture included one convolution layer with 6 5x5 filters with a stride of 1, one 2x2 max pooling layer, three fully connected layers, and three outputs corresponding to the labels *left*, *center*, and *right*. Intuitively the convolutional layers performed feature extraction while the fully connected layers carried the control, however it was not possible to draw a definite line between the two tasks. The first dataset was comprised of 45 minutes of 40x40 pixel smart phone images, down-sampled to 10 FPS in grayscale. 80% of this data was used for training, and testing on the remaining 20% produced an accuracy of 79%. The second dataset consisted of 35 minutes of ATIS output data that was down-sampled into 64x64 pixel frames

sampled uniformly at 10 FPS. The ATIS output polarity was used to create a three value color image in MATLAB, akin to RGB. This dataset was again trained on 80% of the data and tested on 20% which produced an accuracy of 81%.

Figure 4.11 shows an input image taken from the test set and pushed through the trained network. The bottom of the figure depicts driving input and predicted commands, and the sub-figures on the right illustrate the output of the six filters from the convolutional layer. Intuitively it can be seen that the network is carrying out edge detection. In this particular image, the user is driving the robot left towards the wall and the CNN is predicting a correction towards the right. During data collection the robot was randomly steered toward the wall and then corrected to increase the variability of the data and to teach the robot to recover from mistakes.

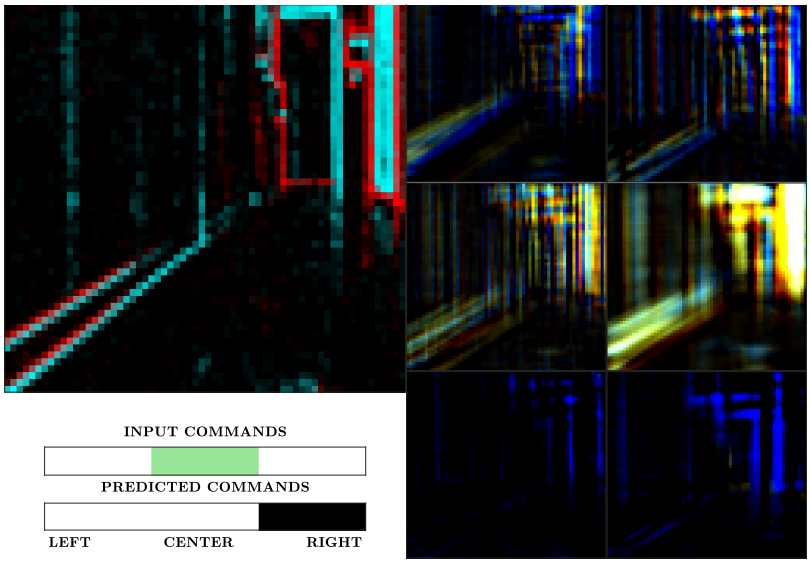


Figure 4.11: MATLAB simulation results for CNN tested on ATIS frames. Note the output of the six convolutional filters appear to be performing edge detection. Moreover, while the driver is aiming at the wall, the predicted command indicates a correction towards the right.

4.4.7.2 Simulation Accuracy

After showing the viability of this CNN approach in Section 4.4.7.1 a separate network was trained in Eedn (see Section 4.4.6) for use on the TrueNorth hardware. The network that gave the highest accuracy on the development datasets contained 11 layers. This network is shown in Figure 4.12. The first layer is a data layer which sends information into the system. Next, there are three sets of layers, each set containing three individual layers. The first set contains two pooling layers followed by a convolutional layer. Each of the second and third sets contain a pooling layer followed by two convolutional layers. The last standard layer is the prediction layer that chooses the direction the robot should be steered, and finally there is a single loss layer (not pictured in Figure 4.12) that Eedn uses to train the network.

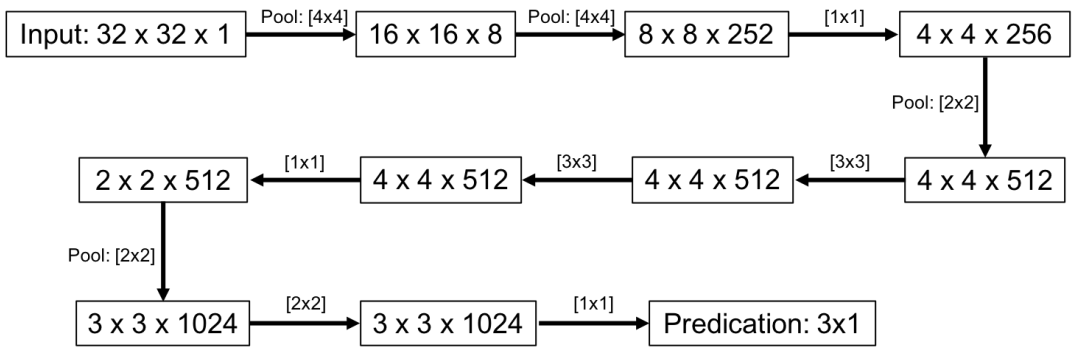


Figure 4.12: Convolutional Neural Network implemented on the TrueNorth hardware and trained in IBM's Eedn framework. Each block shows the size of the output at each layer and the connections between layers are labeled with the size of the kernel used in brackets.

This network was trained for 200,000 iterations on each of the three datasets.

When trained on the residential sidewalk dataset, this network yielded a training accuracy of 95%, a development accuracy of 81%, and an accuracy of 82% when run directly on the TrueNorth hardware. The hallway dataset resulted in a training accuracy of 93%, a development accuracy of 67%, and an accuracy of 69% on the TrueNorth hardware. The campus sidewalk dataset gave a training accuracy of 96%, a development accuracy of 90%, and an accuracy of 87% on the TrueNorth hardware. The accuracy on the TrueNorth hardware and the development data differ due to the constraints placed on weight values in the actual hardware. The training and development curves for each dataset can be seen in Figure 4.13.

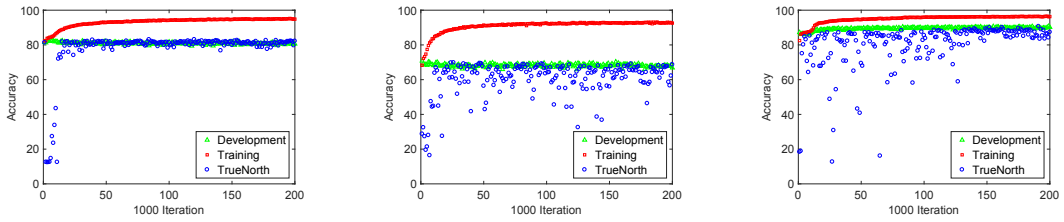


Figure 4.13: Results from training a deep neural network in IBM's Eedn software framework using the (a) residential sidewalk dataset, (b) hallway dataset, and (c) campus sidewalk dataset.

4.4.8 Conclusion

This work illustrated the construction of a fully neuromorphic autonomous robotic driving platform including a spike-based vision system and a spiking "brain" that successfully processes that sensory information. Even with limited training data the

robot determined the correct direction to steer up to 82% of the time depending on the dataset. These results are expected to improve further with more training data and extended training, and these results will be fed into successful real-time implementation.

Chapter 5

Neural Modeling on Neuromorphic Hardware

5.1 Introduction

The field of neuromorphic engineering began in 1989 under the guidance of Carver Mead.[165] Mead first proposed modeling biological neurons by using transistors to represent their behavior within VLSI (Very Large Scale Integration) circuits. From early work within the field of neuromorphic engineering to today, researchers have implemented many different types of VLSI neuron models.[154] By implementing these neurons directly in hardware, many of the complex aspects of neural software modeling are avoided, and transistor characteristics are leveraged for efficient design. From a computational perspective, neuromorphic hardware engineering has also provided

the promise of low-power, general-use computational platforms with faster processing, borrowing inspiration from the brain's own structure and efficiency.

As technology has progressed and the size of the transistor has decreased, neuromorphic engineers have been able to increase the size of neuromorphic chips and therefore the number of neurons a chip can contain; however, there is a tradeoff between the number of neurons and the biological plausibility of said neurons for a given chip size. As Izhikevich summarizes, there have been many computational models of neurons created, each containing varying levels of biological plausibility.[166,167] However, Izhikevich identifies that as biological plausibility increases, computational neurons take increasingly longer to compute and often consume more power, which can be undesirable depending on the application. As the field of neuromorphic engineering grew, so did the range of systems built within it. Neuromorphic processors went from containing just a few neurons to large arrays of neurons, implemented in both digital and mixed-mode circuitry.[17–20,154] In some cases, neuron models were simplified for computational tradeoffs, and some of the biology behind the initial inspiration was abstracted away.[13]

In 2014, IBM announced the release of the TrueNorth neurosynaptic processor.[13] TrueNorth contains one million neurons and provided the first commercially produced neuromorphic platform. With one million neurons per chip, TrueNorth provided a customizable massively-parallel neuron array designed for users outside of the research group that created it. IBM provided a programming environment[168] to accompany

the hardware and thus removed the need to understand the underlying hardware to build applications. TrueNorth consists of a densely-packed array of neurons to achieve one million neurons per chip but also contains architecture tradeoffs to attain its high neuron density.

Previous to the creation of TrueNorth, a number of neuromorphic processors had been developed. One of the first larger-scale neuromorphic processors developed was the spiking neural network architecture SpiNNaker project.[16,169] SpiNNaker was designed to simulate billions of neurons in real time. SpiNNaker consists of 18 ARM968 cores per chip, with different boards containing different numbers of chips, and uses packet communication between cores. Biological relevance was a priority when designing this hardware. Since the development of TrueNorth, two other neuromorphic processors that also contain a large number of neurons have been fabricated: Intel's Loihi processor[14] and Stanford University's Braindrop,[170] created by the Brains in Silicon research group. Both of these processors provide significant deviations from many of the design decisions undertaken by the TrueNorth development team. This chapter provides an overview of the TrueNorth, Loihi, Braindrop, and SpiNNaker neuromorphic processors and an assessment of the utility of each processor for neural modeling and spike-based computations.

5.2 Executing the NEF on Neuromorphic Hardware

As discussed in Chapter 3 Section 3.2.3, the Neural Engineering Framework (NEF)[28,29] provides a framework for describing neural populations and the connections between them mathematically. Nengo[138] is a Python library, built upon the NEF, that allows users to describe NEF-based models in code and then simulate the the models on different platforms. The NEF and Nengo can be used both to explore spiking-neuron based computation as well as for neural modeling.

Nengo simulations are most often run on a generic CPU, but backends have been developed to map the framework for use with GPUs[138,171] and field programmable gate arrays (FPGAs).[172] Nengo simulations can also be run on existing neuromorphic hardware platforms. Nengo simulations can be run on the SpiNNaker chip,[16,27,173] Neurogrid,[174,175] and a mixed signal neuromorphic multi-neuron VLSI chip.[176] Recent work has also extended the NEF to non-ideal silicon synapses[177] through the creation of Braindrop,[170] a mixed-signal processor that leverages the NEF to exploit the variability inherent in analog sub-threshold transistors. These neuromorphic implementations attempt to provide faster, lower-power platforms for neural modeling. The following subsections will describe the execution of Nengo models on Braindrop as published by the Braindrop team,[170] work by the author to execute those same models on TrueNorth and the backend developed to do so, and the execution of the

same models on SpiNNaker and Loihi.

5.3 Neural Modeling on Braindrop

5.3.1 Overview of Braindrop

Braindrop[170] is a mixed-signal neuromorphic processor designed specifically to efficiently execute Nengo models with minimal power consumption. Braindrop leverages the inherent non-uniformity of analog design in its mapping and execution of Nengo models. Braindrop contains 4096 neurons, 64 KB of weight memory, 1024 accumulator buckets, 1024 synaptic filters, and 2 small memories for a Pool Action Table (PAT) and a Tag Action Table (TAG) to store other parameters. The PAT has 64 entries, enabling a pool-size granularity of 64 neurons. TAT has 2048 entries for redirecting tags to a subset of accumulator buckets, synaptic filters, and other cores. Braindrop was implemented in a 28-nm FDSOI process and consumes 381 fJ per equivalent synaptic operation for typical network configurations.

There are two main aspects of the Braindrop design that enable its efficient execution. First, Braindrop decodes NEF models by accumulative thinning. Accumulative thinning computes a linearly weighted sum of spike rates to reduce the number of spikes or deltas that must be fanned out in connections. This operation reduces a $N \times d \times d \times N$ decode from $O(N^2d^2)$ spike traffic to $O(Nd)$. This greatly reduces both energy and time. Second, Braindrop employs sparse encoding by spatial convolution

to calculate the encodes for each neural population. This is accomplished by using a diffuser operation to convolve current from specific "tap-points" through a resistive grid of transistors to create a distributed range of encoders for the population.

In addition to the hardware design, the Braindrop team also created an associated software stack. This calculates, based on the Nengo model to be mapped to the hardware, where to place neuron "pools" on chip, among other tasks. It also accomplishes the Nengo backend, such that the user experience for Braindrop mirrors that of other neuromorphic hardware platforms with a Nengo backend (with the exception of TrueNorth as described in Section 5.4).

5.3.2 Modeling Results

As detailed in their paper,[170] the creators of Braindrop implemented a number of Nengo models on their hardware for initial validation. Figure 5.1 shows one set of such results, reprinted with author permission, mapping a one-dimensional neural population onto the Braindrop hardware. Here neural populations of 256 and 1024 neurons were used to implement a sinusoidal transform, $y_f(x) = F_{max}(0.5+sin(f\pi x))$ where $f \in 1, 2, 4$. F_{max} is equal to 500 Hz (orange), 1000 Hz (green), and 1500 Hz (blue). This figure also shows histograms of the calculated decoded weights for this functional approximation.

For comparison, Figure 5.2 shows the same results run on the CPU of a 2015 Mac Book Pro with a 2.8 GHz Intel Core i7 processor at 2.8 GHz with 16 GB of

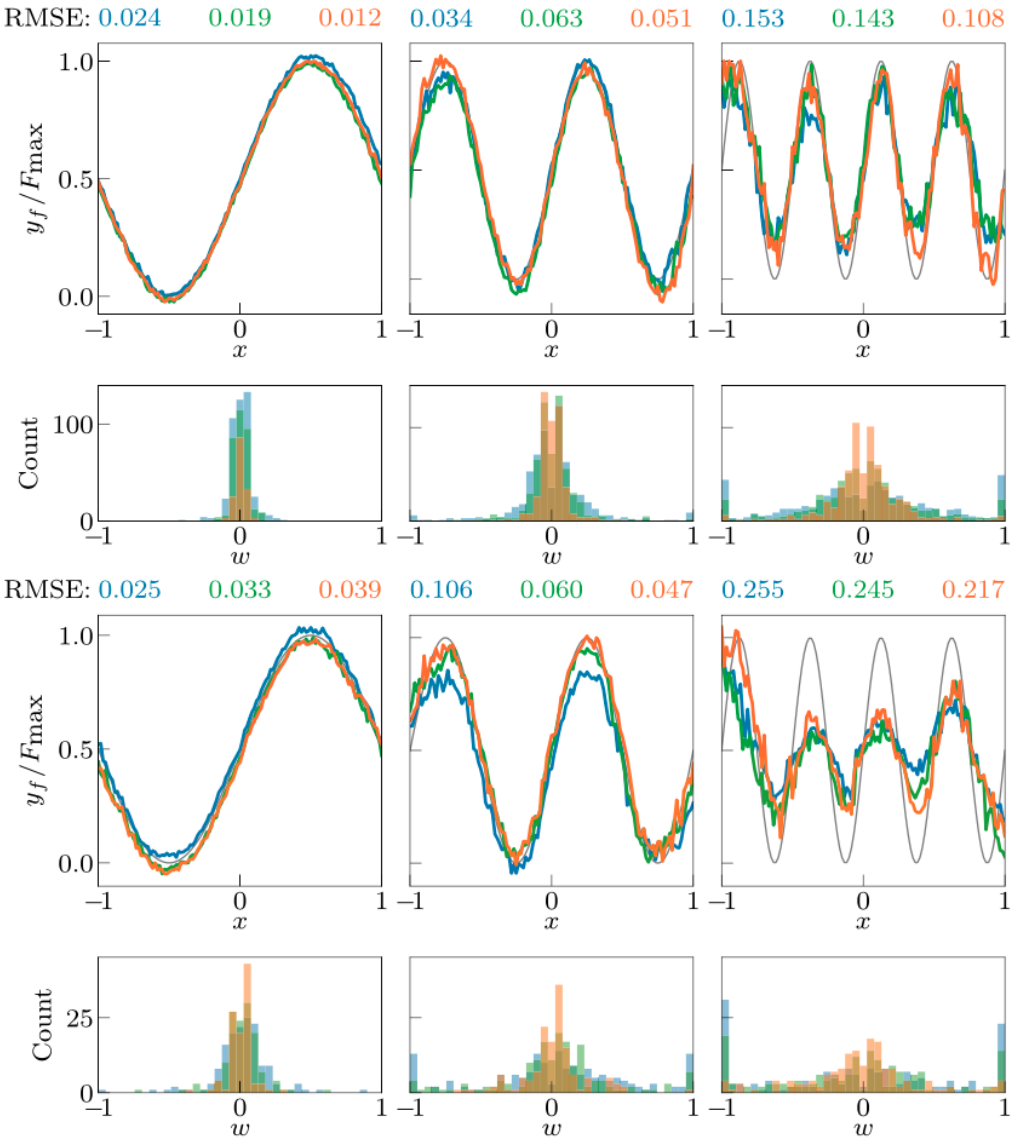


Figure 5.1: Decoded output from a Nengo model executed on Braindrop. The top row shows decoded output for sinusoidal functions of increasing frequency using a neural population of 1024 neurons. Below it are the histogram of weights required for the decodes of the 1025 neural population. The third row from the top are the decoded outputs for sinusoidal functions of increasing frequency using a neural population of 256 neurons. The bottom row shows a histogram of weights required for decode of 256 neural population. Figure reprinted with author permission[170]

memory. This figure highlights the increased accuracy of representing given larger neural populations.

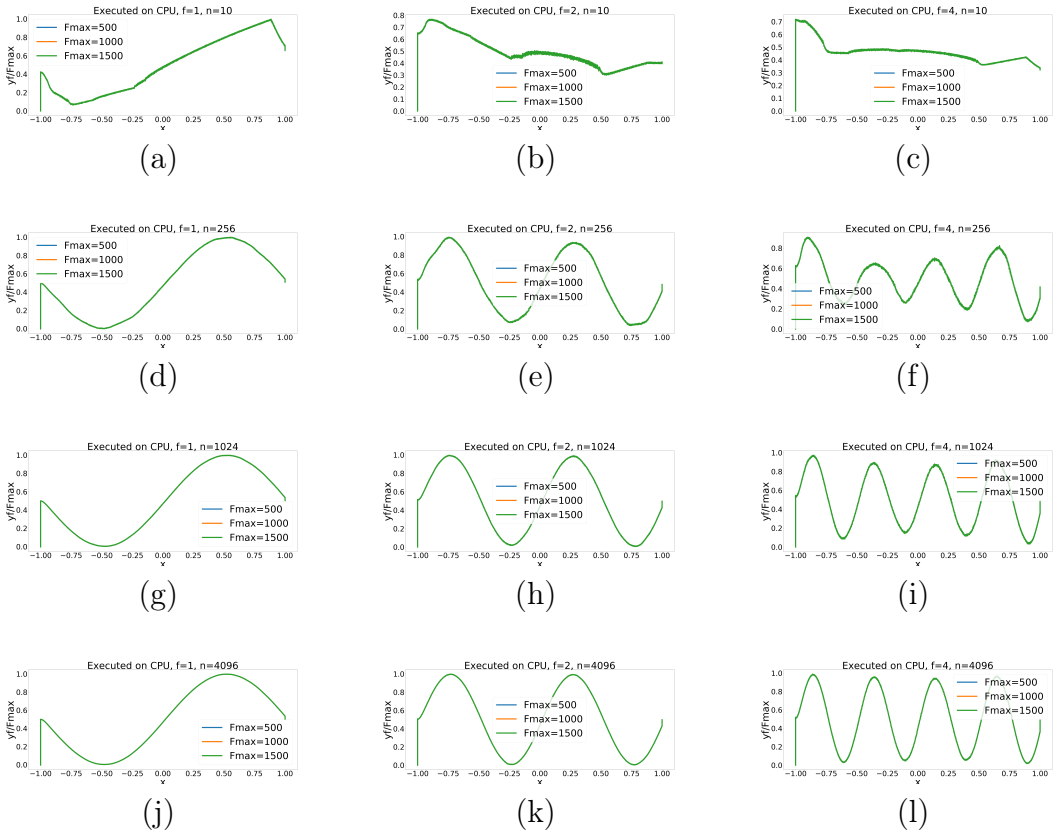


Figure 5.2: Illustrates the execution of a one population Nengo model on a CPU performing a sinusoidal transformation on its input x. For all models the neuron population computes $y_f(x) = F_{max}(0.5 + sin(f\pi x))$ where $f \in 1, 2, 4$. F_{max} is equal to 500 (orange), 1000 (green), and 1500 (blue). (a-c) Illustrate the decoded output using a 10 neuron neural population. (d-f) Illustrate the decoded output using a 256 neuron neural population. (g-i) Illustrate the decoded output using a 1024 neuron neural population. (j-l) Illustrate the decoded output using a 4096 neuron neural population. As expected, the outputs for $F_{max} = 500$, $F_{max} = 1000$, and $F_{max} = 1500$ match when normalized.

5.4 Neural Modeling on TrueNorth

TrueNorth is a multiprocessor with a tightly coupled processor/memory architecture that results in energy efficient neurocomputing[13] *. It is comprised of 4096 cores, each core with 65 K of local memory (6T SRAM), "synapses", and an Arithmetic Logic Unit (ALU) that computes 256 integrated and fire "neurons" using 20 bit fixed point arithmetic. The cores are event-driven using custom asynchronous and synchronous logic and are globally connected through an asynchronous packet switched mesh network on chip (NOC). The chip development board includes a Xilinx Zynq 7000 SoC (system on chip) that performs housekeeping and provides standard communication support through an Ethernet UDP interface. A hierarchical, compositional programming language is available to develop on-chip applications.[31] IBM provides support and a development system as well as "Compass", a scalable simulator.[149]

Each TrueNorth neuron has a number of parameters including synaptic weights, a membrane potential, a spiking threshold, and a delay allowing each neuron or a small group of neurons to take on a wide variety of behaviors.[178,179] All cores operate in parallel, with spikes traveling from neurons to destination axons every "tick", designated by a global synchronization clock usually running at 1 kHz.

*Parts of this subsection, and earlier versions of this subsection were previously published by the author.[146]

5.4.1 TrueNorth Corelets to Execute the NEF

To implement the NEF using Rectified Linear Unit (ReLU) neurons on TrueNorth, corelets were designed to perform the following equations:

$$J_n = \sum_i ((e_{in} x_i) g_n + b_n) \tag{5.1}$$

$$s_n = floor[max(J_n/T, 0)] \tag{5.2}$$

$$z_k = \sum (s_n d_{nk}) \tag{5.3}$$

Equation 5.1 describes the input current J to each neuron n in the given population, using the input value x, encoding matrix e, gain g, and bias b. Equation 5.2 states the conditions under which each neuron emits s spikes, following a ReLU neuron equation, where T is the threshold voltage. Equation 5.3 gives the decoded value z that the neural population represents, given the decoding matrix d. In this implementation the values for the encoder, bias, gain, and decoder matrices are computed using Nengo, and then scaled and rounded to integer values for mapping to the TrueNorth hardware. The corelets created to implement the calculations in these equations are described in the following subsections.

5.4.1.1 Vector Matrix Multiplication

The TrueNorth architecture provides a lookup table of four weights per neuron column of its crossbar array, which presents challenges for calculations requiring high bit precision multiplication. Previous work existed which described a method for 4-bit vector matrix multiplications on TrueNorth.[180] That work was extended to produce a corelet which performed 9-bit signed multiplication on TrueNorth.[181] The 9-bit signed multiplication corelet was used with author permission in this work to map NEF calculations onto TrueNorth.[181] The precision was vital for accurate mappings of NEF models onto TrueNorth. The vector matrix multiplication corelet was used to multiply the input vector and encoder matrix in Equation 5.1 and the output spike vector and decoding matrix in 5.3.

5.4.1.2 Addition and Vector Multiplication

To minimize time and resources, the addition and multiplication in Equation 5.1 was combined into one corelet. Gains are programmed as synaptic weights and applied to the previously calculated product of the input value and encoding matrix. That product is then added to the bias, all within this corelet. As seen in Figure 5.3, each input is decomposed into a positive part and a negative part because on TrueNorth there is no way to differentiate "positive" and "negative" spikes. The "positive" input is denoted by x_{1+} for input x_1, and x_{2+} for x_2 and the "negative" input is denoted by x_{1-} for input x_1, and x_{2-} for x_2. The bias is introduced into the crossbar by a bias trigger

signal which is sent into the crossbar array at the beginning of each computation time window. The one-bit bias trigger is initially multiplied by a synaptic weight equal to the corresponding NEF neuron's bias and then fed back into the crossbar to be multiplied by a scaling value, v, before it is added to the "positive" or "negative" neuron's membrane potential. If the bias is positive it is added to the "positive" neuron's membrane potential and if it is "negative" it is added to the "negative" neuron's membrane potential. Thus the "positive" and "negative" neurons will give the result of Equation 5.1 when aggregated together. Although the bias for each neuron was initially passed into each crossbar as another input, this corelet was later changed to the described design to reduce the overall number of inputs that would need to be passed into the system, especially since the number of bias inputs would grow proportionally to the size of the neuron population in the NEF model. The corelet behind the initiation of the bias trigger is the same as the corelet described in Subsection 5.4.1.5.

5.4.1.3 Rectified Linear Neuron Unit

In this work a Rectified Linear Unit (ReLU) neuron model was utilized as the neuron model for the NEF/Nengo neurons. These neurons fire according to Equation 5.2. Nengo models can be implemented using any neuron model, but because of the constraints of the TrueNorth architecture, a ReLU neuron provided an achievable neural computation given the digital platform and neuron parameters of TrueNorth.

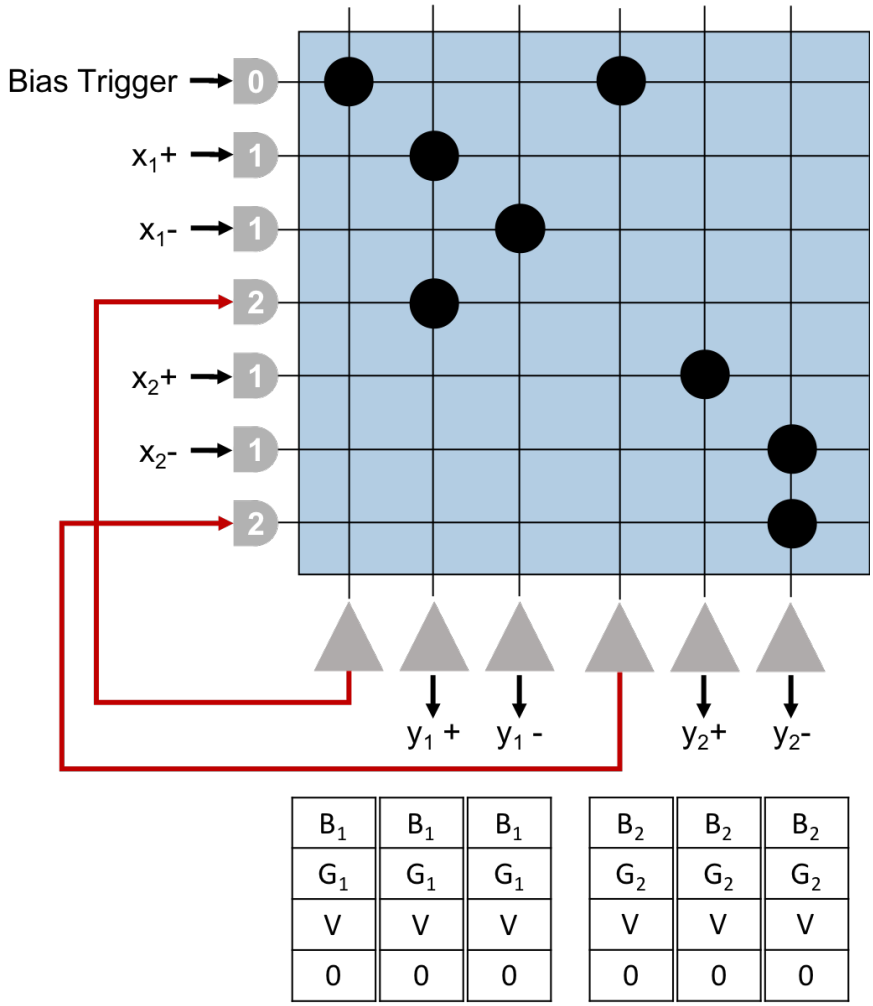


Figure 5.3: This corelet calculates the multiplication of each NEF neuron's gain and the addition of its bias. The previously calculated product of the input and encoding matrix flow into the crossbar, separated into "positive" and "negative" parts, each entering through their designated axon. They are multiplied by a synaptic weight equal to the gain and aggregated in their corresponding "positive" or "negative" neuron. The bias signal is initiated by a bias trigger at the beginning of the computation window. This one-bit trigger is multiplied first by the bias of that neuron, and then fed back to also be multiplied by a scaling value, v, before being aggregated into the correct sign neuron's membrane potential.

As previously stated, the inputs to each Nengo neuron are divided into a positive and negative component and sent to the crossbar array in parallel. Concurrent with the

arrival of the inputs, a trigger spike is sent into this core with a synaptic weight of -255, initializing the neuron's membrane potential to -255. Once all input spikes have been received by the Nengo neuron core, a second trigger is sent into the core, increasing the membrane potential by 255 and consequently setting the membrane potential to the resultant summation of the negative and positive inputs. If values fall below zero, the membrane potential will not reach the neuron's threshold; therefore spikes will not emit from the neuron, resulting in correct rectified linear unit computation. This neuron has a threshold of T, as seen in Equation 5.2, causing the number of spikes emitted by those neurons to be equal to s. If the calculations produce values larger than 255, multiple triggers can be used to provide a proportional negative initialization.

This calculation initialization is necessary because without it there is a chance of calculating the wrong result. For example, if the positive component of the input arrives into the corelet first, it could cause the neurons to achieve membrane potentials above their threshold, when in reality the final value will not be above threshold due to the negative component of the input. Through initialization of the membrane potential, the order of the incoming spikes have no bearing on the output spikes and the correct values are computed.

5.4.1.4 Neuron Membrane Reset

After the arrival of the second trigger to the ReLU neuron corelet, each ReLU neuron will emit its spikes while simultaneously dropping its membrane potential to zero, following the TrueNorth "linear" neuron model. However, if the ReLU neuron has a negative membrane potential after the second trigger, it will not emit any spikes and will maintain a negative membrane potential. Its membrane potential must be reset to zero before the computation window begins for the next computation window to produce a correct calculation. The ReLU corelet shown in Figure 5.4a provides this added functionality. Here, three neurons are used. Neurons 0 and 2 are identical, whereas Neuron 1's membrane potential will be equal to Neuron 0's potential multiplied by -1. For example, if after the end of the second trigger Neuron 0 has a membrane potential of -5, Neuron 1 will have a membrane potential of $+5$ causing it to send 5 spikes to Neuron 0 thereby increasing its membrane potential back to zero and effectively resetting the membrane potential. Alternatively, if Neuron 0 has a positive potential, Neuron 1 will not send it any spikes but Neuron 2 will send Neuron 1 the appropriate number of spikes to reset Neuron 1's potential to zero. By adding this functionality, the ReLU neuron can be used to perform continuous calculations and execute actual NEF simulations on TrueNorth.

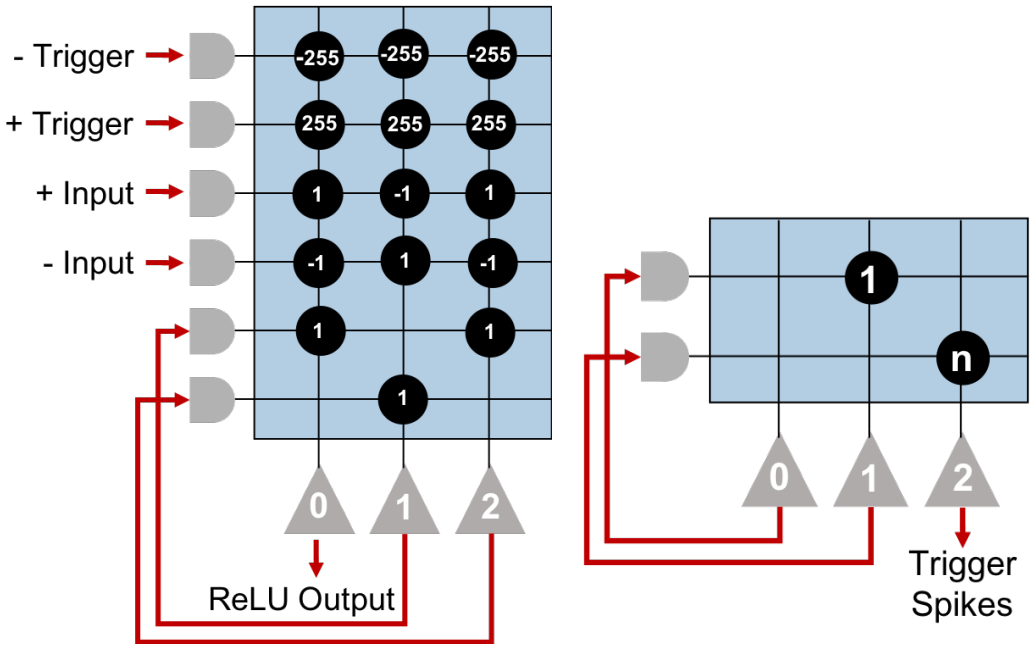


Figure 5.4: (a) This corelet implements the ReLU neuron. First the negative trigger initializes all neurons -255. Then the positive and negative components flow in and are aggregated by the output neurons. Lastly, the positive trigger increases the membrane potentials by 255, allowing the output neurons to become equal to their true output value. This corelet also implements the membrane potential reset needed to enable actual NEF simulations to execute on TrueNorth in successive. By feeding the outputs back from Neurons 1 and 2, it allows the membrane potentials of all 3 neurons to reset to zero after each calculation. (b) This corelet implements a clock for the trigger functionality needed to provide the initial decrease in membrane potential and the final increase in membrane potential for the ReLU neurons. This corelet uses three neuron types. Type 1 has a leak of $+1$ and a threshold of d, a large integer value. Type 2 has no leak and a threshold of T/d where T is the ideal trigger time. Type 3 has no leak and a threshold of 1. Because the first neuron has a leak of $+1$ it does not require input to spike. Neuron 2 outputs n spikes beginning at time $ceil(T/d) * d$.

5.4.1.5 Trigger Clock Signal Generation

The trigger corelet consists of three neuron types (indicated in Figure 5.4b) on each of the neurons. Neuron type 0 has a leak of $+1$ and a threshold of d, a large positive

value. This neuron creates the base clock signal by spiking every tick. Neuron type 1 has no leak, and a threshold of T/d, where T is the time in ticks that the trigger should be sent to the ReLU neuron. Neuron type 2 also has no leak and has a threshold of 1. With these neurons the trigger occurs at tick $ceil(T/d) * d$.

Because the first neuron has a leak of +1 this neuron does not need an input to spike. As well, the synaptic weight which connects to the neuron of type 2, has a weight of n where n is the number of triggers fed into the ReLU neuron. Because synaptic weights can only be in the range $[-255, 255]$, to create an initialization value of greater magnitude, we use multiple spikes in series. This trigger corelet was modified from the trigger corelet described in previous work.[181]

Since TrueNorth can only implement integer thresholds, the values of T and d can be manipulated to produce a trigger time close to the desired trigger time. This is not problematic for the implementation of the NEF, as long as the trigger fires after all incoming spikes have arrived at the ReLU neuron. If this is observed, accuracy will not be affected, although run-time may be slightly longer than necessary to produce correct computations.

5.4.2 Results of NEF Mapping onto TrueNorth

This work requires only six corelets per neural population to map the NEF onto the TrueNorth hardware to execute NEF simulations. These corelets vary in their implementation size depending on the number of neurons in the represented Nengo

neural population and the number of dimensions that population represents.

5.4.2.1 Resources Needed

For this implementation the largest neural populations implemented on a single TrueNorth chip contain 11,789, 10,009, and 6,880 neurons representing 5, 10, and 30 dimensions respectively, as seen in Figure 5.5. Populations representing higher dimensions require more resources. Thus, populations that represent more dimensions must contain fewer neurons to fit on one TrueNorth chip. However, one can tile TrueNorth chips in series to run even larger neural populations.[182]

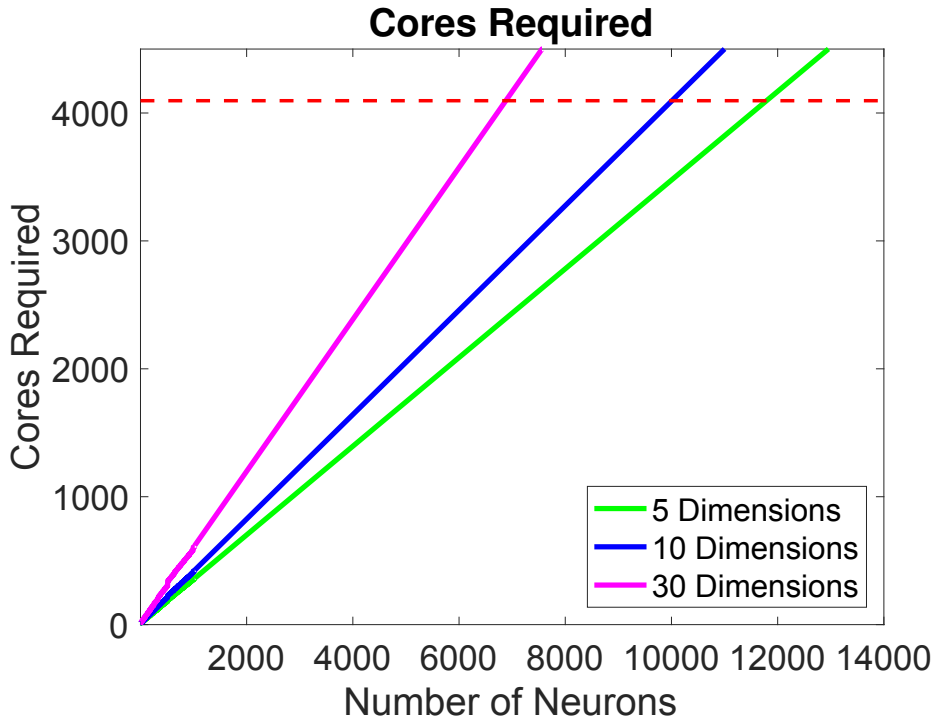


Figure 5.5: The relationship between neural population size and the number of cores required. The maximum sized neural populations that can fit on one TrueNorth chip are 11,789, 10,009, and 6,880 neurons representing 5, 10, and 30 dimensions, respectively.

5.4.2.2 Computation Time

The computation time to execute NEF models on TrueNorth is directly dependent on the computation window necessary for all spikes to flow through all cores. This window is a linear function of the maximum value used in the computations. This maximum value is also what determines the trigger times for the ReLU neurons, which must be contained within the window for a single computation.

5.4.2.3 Accuracy

To implement NEF and Nengo models on TrueNorth all encoder, decoder, gain, and bias matrices must be rounded and scaled to integer values within the appropriate ranges for the hardware. This TrueNorth mapping matches the results calculated in Python with the rounded values. 100 trials were executed with a neuron population of 1000 neurons, representing one dimension. Their averaged results after rounding produce a root mean squared error of 1.06.

In addition to error introduced by rounding, there is also error introduced during the division of the threshold in the ReLU corelet. Because all values are represented with a "positive" component and a "negative" component, when the ReLU performs that division, sometimes the resulting value is off by one due to the integer division. Although not initially an issue, this off-by-one error can create much larger errors later when that value is next multiplied by the decoding matrix. As this error propagates it makes it difficult to attain any accuracy beyond a single neuron population model.

Future work could explore altering the VMM corelet to help mitigate this issue.

5.4.2.4 Modeling Output

Figure 5.6 replicates the neural simulation results shown in Figure 5.1 from neural simulation execution on Braindrop and the results in Figure 5.2 from neural simulation execution on a CPU, on TrueNorth. As shown by Figure 5.6, there is error introduced during the NEF transformation, mostly due to the rounding to integer weights required by TrueNorth, as well as an error introduced due to the division in the ReLU neuron equation.

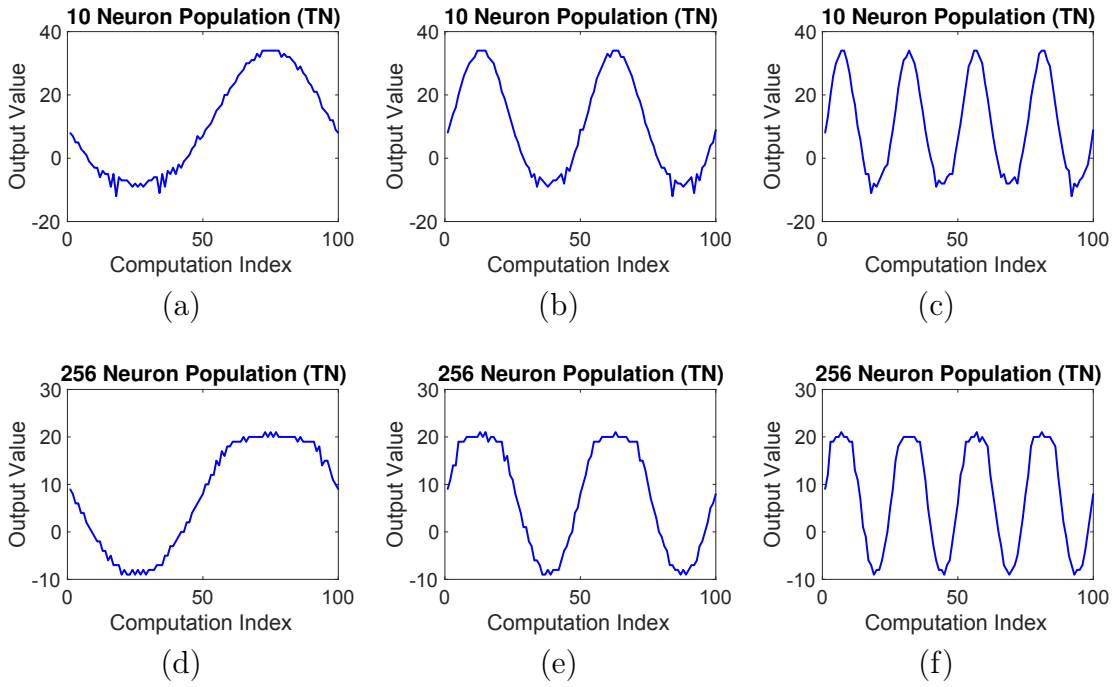


Figure 5.6: Illustrates the execution of a one population Nengo model on TrueNorth performing an identity transformation on its input x. For all models the neuron population computes the identify of an input equal to $y_f(x) = F_{max}(sin(f\pi x))$ where $f \in 1, 2, 4$. F_{max} is equal to 30. (a-c) Illustrate the decoded output using a 10 neuron neural population. (d-f) Illustrate the decoded output using a 256 neuron neural population.

5.5 Neural Modeling on SpiNNaker

5.5.1 Overview of SpiNNaker

The spiking neural network architecture (SpiNNaker) project began with the goal to create a processor that could simulate billions of neurons in real time.[16,169] A SpiNNaker machine consists of 48 SpiNNaker nodes connected on a printed circuit board (PCB), shown in Figure 5.7. Each SpiNNaker node is a package that contains a custom multiprocessor system-on-chip integrated circuit that includes 18 ARM968

processors, each with their own local 32 kB instruction memory and 64 kB data memory. These processors are connected via a network-on-chip (NOC) to each other, as well as to shared on-chip resources, and a 128 MB low-power mobile dual-data-rate (DDR) DRAM.

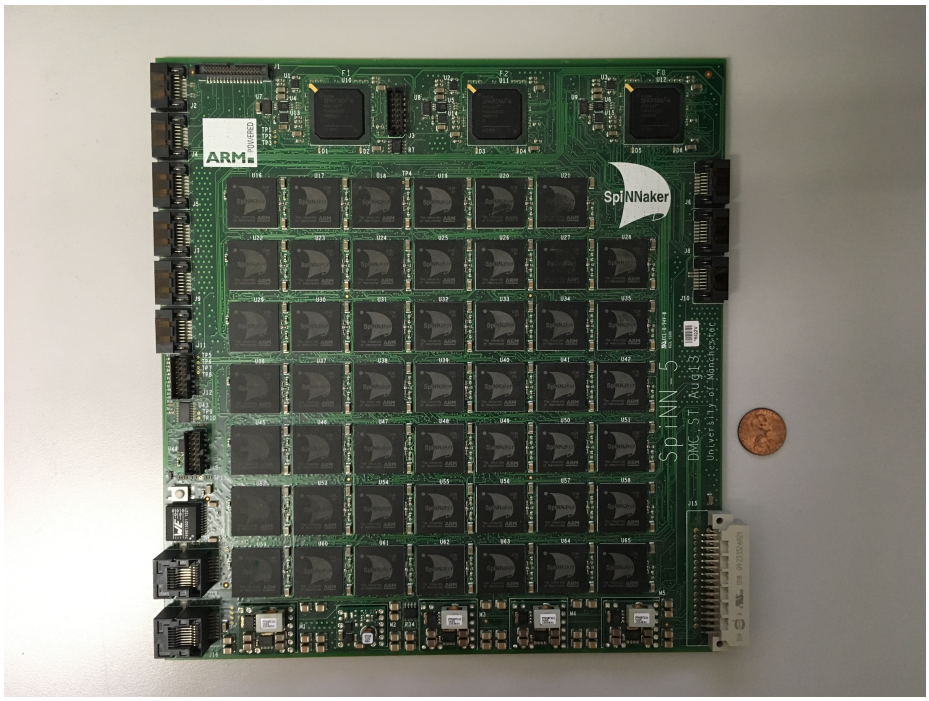


Figure 5.7: A 103 SpiNNaker machine, containing 48-nodes and 864 ARM processor cores. It requires a 12 V, 6 A supply, and uses two 100 Mbps Ethernet connections, one for the Board Management Processor and one for the SpiNNaker array. 103 boards can be connected together to form larger systems.

Primarily the cores communicate with each other through packets. These packets follow the concept of Address Event Representation (AER) and convey a spike's time and the neuron from which it originated. Although SpiNNaker employs a two-dimensional physical communication structure, the underlying architecture of the system allows for multi-dimensional networks to be easily mapped onto the hardware.

Packets can be 40 to 72 bits long and are directed by a router. Packets can either be sent to a neuron's nearest neighbors, point-to-point, multicast, or sent to a fixed route.

SpiNNaker is optimized for modeling complex networks of simple point neuron models. Biological relevance was highly considered while the system was designed. Given the human brain consists of 10^{11} neurons, SpiNNaker was designed to model 1% to scale, or be able to model a billion neurons. To date SpiNNaker has been used in many projects involving hardware achitecture and design, neural simulation and system software, and neural engineering.[183]

5.5.2 Modeling Results

As previously described for TrueNorth, the same neural model was run on a 103 SpiNNaker machine (pictured in Figure 5.7). Figure 5.8 shows these results for neural populations containing 256 and 1024 neurons. As expected, the output corresponds to the output produced by the models executed on TrueNorth.

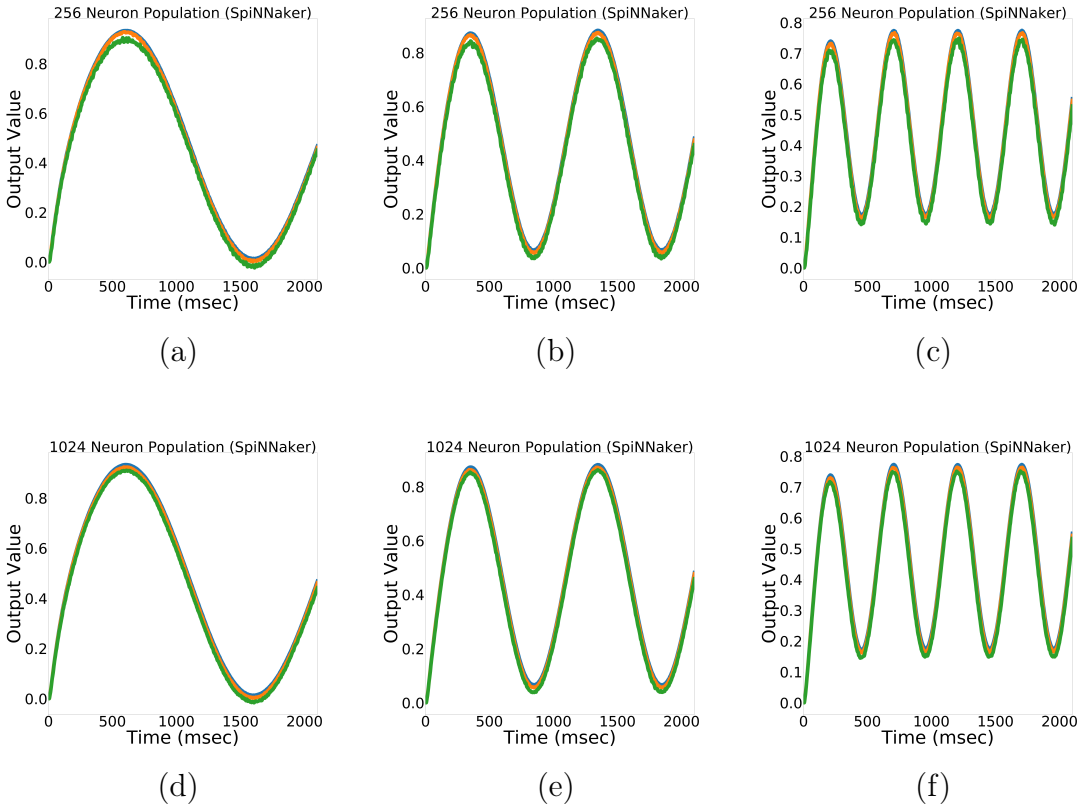


Figure 5.8: Illustrates the execution of a one population Nengo model on a SpiNNaker performing an identity transformation on its input x. For all models the neuron population takes as input $y_f(x) = F_{max}(0.5 + 0.5 * sin(f\pi x))$ where $f \in 1, 2, 4$. F_{max} is equal to 1 (orange), 0.5 (green), and 0.25 (blue). (a-c) Illustrate the decoded output using a 256 neuron neural population. (d-f) Illustrate the decoded output using a 1025 neuron neural population.

5.6 Neural Modeling on Loihi

5.6.1 Overview of Loihi

Loihi,[14] designed by Intel in 2018, is a $60mm^2$ chip neuromorphic processor fabricated in Intel's 14 nm FinFET processes, that allows for on-chip leaning and other features not previously realized on neuromorphic hardware. Loihi implements 2.07

billion transistors and 33 MB of SRAM. Its digital architecture contains 128 neuromorphic cores, each with 1,024 primitive spiking neural units or compartments. Loihi cores contain four main unit types: 1) Synapse units process incoming spikes and read out the associated synaptic weights from memory, 2) Dendrite units update the synaptic current, u, and membrane potential, v, for all neurons within the core, 3) Axon units generate outgoing spikes, and 4) Learning units update synaptic weights according to the given learning rule. In addition to the 128 neuromorphic cores, Loihi contains three embedded x86 processors and off-chip communication interfaces which allow the connection mesh to extend in four directions onto other chips. Loihi contains an asynchronous network-on-chip (NoC) which transmits all communication between cores. Loihi has 16 MB of synaptic memory, providing 2.1 million unique synaptic variables per mm^2.

Special features of Loihi include its sparse network compression, core-to-core multicast communication, variable synaptic formats, and population-based hierarchical connectivity. Additionally Loihi's mesh communication employs a barrier synchronization technique that allows time steps to be dependent on computation time, and not governed by a global synchronization clock. Loihi provides a number of programmable variables to enable different types of on-chip learning.

5.6.2 Modeling Results

Due to limitations in the author's access to Loihi hardware, the author's execution of neural models on Loihi was constricted to during the Telluride 2018 Neuromorphic Cognition Engineering Workshop. During that workshop, a simplified version of the amygdala model described in Chapter 3 was executed on Loihi. Because this workshop was the first public access to Loihi, there were a number of barriers overcome to enable this simulation. These results of the execution of a Nengo neural model on Loihi are shown in Figure 5.9 and Figure 5.10.

5.7 Comparison of Neuromorphic Hardware

In the previous section, the results of simulating a one-population Nengo model on three of the four different neuromorphic platforms were shown. Figure 5.9 and Figure 5.10 illustrate the results of simulating a simplified version of the amygdala model described in Chapter 3 on a CPU, Braindrop, and Loihi, as well as shows neurons with the same response measured in primate amygdalae. Each column of Figure 5.9 and Figure 5.10 illustrate a neuron from a model run on each of those platforms as well as from the primate amygdalae with the same response to the same experiment. The differences in the plots illustrate some of the differences of the hardware platforms.

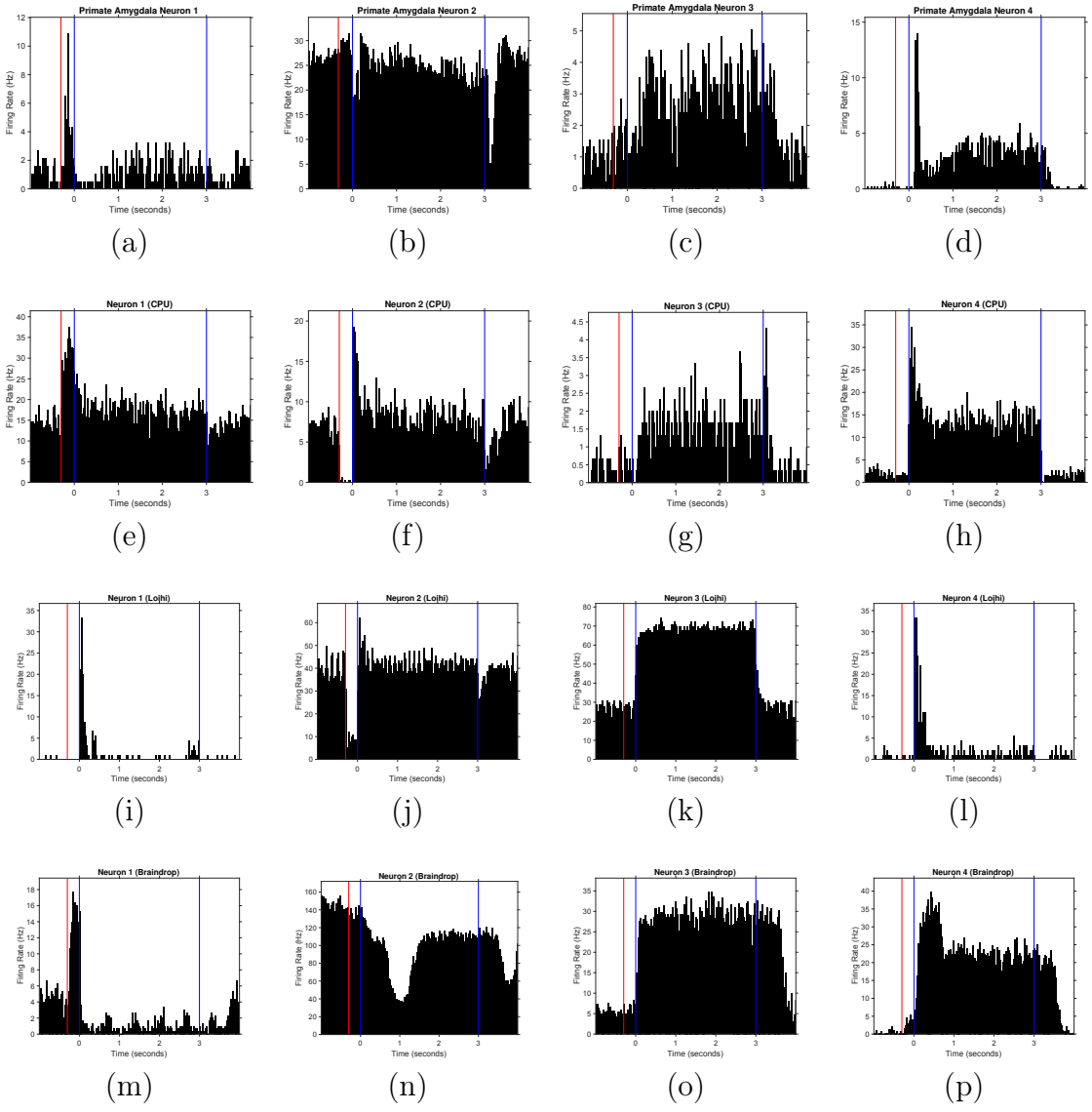


Figure 5.9: Illustrates neuron responses measured from an early version of the model described in Chapter 3. Each column contains neurons measured from a different platform that give the same response to the experiment. (a-d) Neurons measured from primate amygdalae. (e-h) Neurons measured from a model executed on a CPU. (i-l) Neurons measured from a model executed on Loihi. (m-p) Neurons measured from a model executed on Braindrop.

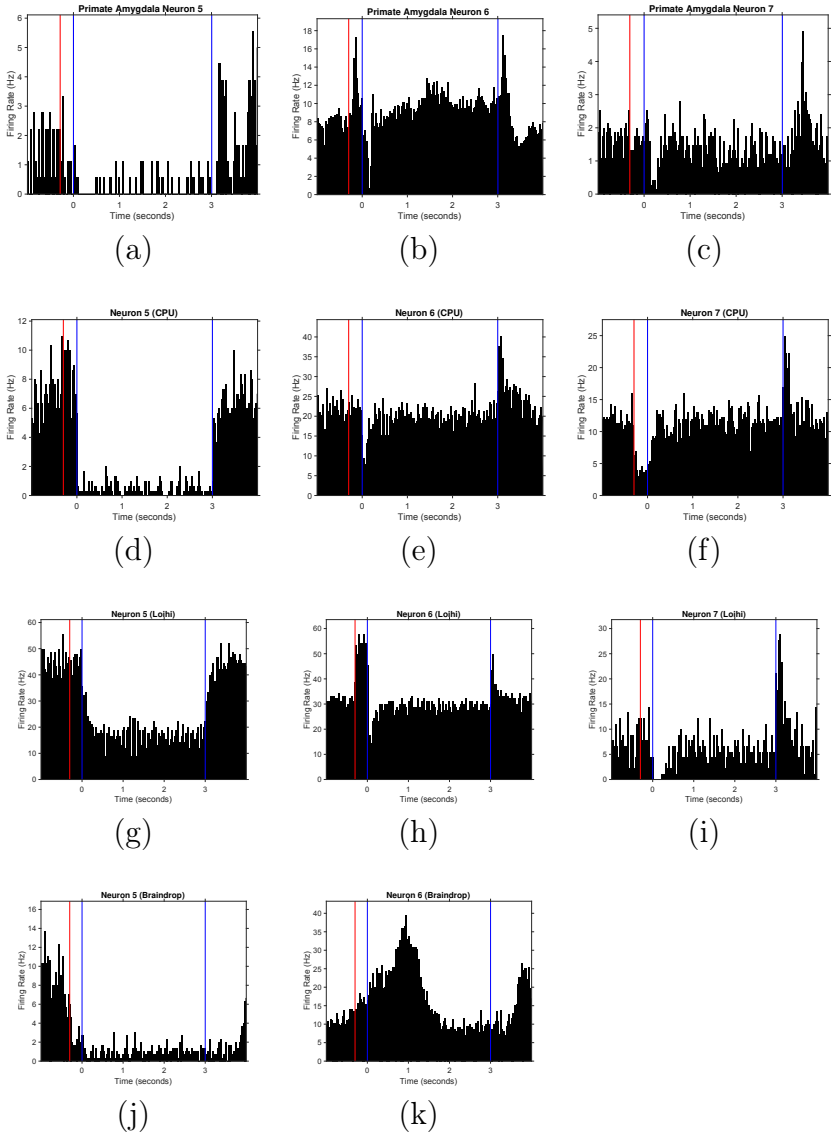


Figure 5.10: Illustrates additional neuron responses measured from an early version of the model described in Chapter 3. Each column contains neurons measured from a different platform that give the same response to the experiment. (a-c) Neurons measured from primate amygdalae. (d-f) Neurons measured from a model executed on a CPU. (g-i) Neurons measured from a model executed on Loihi. (j-k) Neurons measured from a model executed on Braindrop.

5.8 Discussion

Just as each of the discussed neuromorphic hardware platforms was designed under different constraints and for a different purpose, each platform provides different challenges for the execution of neural models. Because Braindrop was specifically designed to efficiency execute Nengo models, it provides the most straightforward user path to do so. Still, given the author's early access to the Braindrop chip, at the time of this experiment there were challenges to overcome with mismatch and timing, as the author was the first Braindrop user outside of the group under which it was created.

Overall TrueNorth is not well suited to execute Nengo models primarily due to its "tick" global clock. Because the backend implementation of Nengo on TrueNorth represents values as a number of spikes within a computation window, iterations of Nengo model computations take a large number of "ticks" to complete. This is not practical for executing large-scale Nengo models, which can already take a long time on traditional CPUs. Additionally, because TrueNorth only provides integer synaptic weights, Nengo weights must be rounded which introduces errors into the Nengo calculations. Lastly, because of the specific implementation of the ReLU neuron corelet, errors are introduced during each neural population calculation and are compounded in subsequent calculations, which are not sustainable for accurate calculations of many population models.

SpiNNaker is the oldest of the neuromorphic platforms included in this work so

its user interface and work flow had previously been streamlined by previous users of the system. Because of that, the author found it most straightforward to execute Nengo models on SpiNNaker. However the Nengo implementation for SpiNNaker does not allow users to record spikes of individual neurons within the Nengo model. It does allow users to record decoded values of those neural populations, but for an application where the individual neuron spikes are needed for analysis, SpiNNaker is not a useful platform.

The author only had access to Loihi at the Telluride 2018 Neuromorphic Cognition Engineering Workshop and at that workshop Loihi was quite cumbersome to work with. Specifically there were still issues loading large amounts of data onto the chip as input and reading off the large output arrays of neuron spikes. As with some of these other neuromorphic processors, Loihi was not designed for users intending to record from every neuron of a Nengo model at once, as is necessary for the neural modeling work described in this thesis.

5.9 Conclusions

In conclusion, although neuromorphic engineering began with notion that transistors could serve as models for biological neurons, the current neuromorphic hardware platforms present challenges for the execution of neural models, specifically models created using the NEF and Nengo. In large neural systems where a complete

connectomics is not known (and even for those where it is), the NEF provides a straightforward approach to building functional models. Nonetheless, both platforms designed specifically to execute NEF models, as well as those that were not, present challenges when simulating these large models. Many of the existing platforms were designed to efficiently execute trained neural networks for machine learning or for spike based computation generally speaking, and specifically not designed for neural model execution. To date, few users, if any, have employed neuromorphic hardware for the actual execution of neural models. As shown in this chapter, neural models can be executed on the existing neuromorphic platforms, but often at a cost such that CPUs and GPUs can still often provide a comparable or even better modeling platform. But given necessary shifts in computing due to the end of Moore's Law, neuromorphic hardware does continue to hold a promise Von Neumann architectures do not.

5.10 Social Robot Neural Model Application

Despite the drawbacks of large scale neural modeling on current neuromorphic hardware, these systems do provide an opportunity to realize novel applications in biological plausible computing. One application area of such computing is social robotics, which requires power-intensive models running on autonomous, portable hardware.

5.10.1 Introduction and Background

Social robotics is a highly useful field; up to now, social robots have been used to provide companionship and elderly care,[184,185] act as receptionists[186,187] and domestic servants,[188] aid in the teaching of children,[189] and assist in research to better understand cognitive conditions that affect social processing like autism[190,191] to name a few [†]. However, the complex algorithms needed to build robots that can successfully interact with humans in a meaningful social way require computational power previously unattainable in mobile systems. Modern low-power embedded platforms, and fields like neuromorphic computing,[9–11] provide solutions to run computationally intensive algorithms on mobile platforms, necessary for robotic social interaction. This section presents a socio-emotional robot with distributed neuromorphic processing. The robot's internal model generates emotional states based on visual input, and then executes predetermined behavior based on the computed internal emotional state. Intel's Loihi,[14] University of Manchester's SpiNNaker,[16,26,27] and Stanford University's Braindrop[15] were each used to run one nuclei of the model on a different neuromorphic processor. Although simplified, this system provides a proof-of-concept using a novel software and hardware processing framework for more emotionally complex social robots.

[†]This section was previously published by the author.[192]

5.10.1.1 Existing Social Robots

Breazeal categorizes the wide range of existing social robots into four main categories: socially evocative, social interface, socially receptive, and sociable.[193] Socially evocative robots are robots where humans attribute social responsiveness to the robot but the robot's behavior does not actually reciprocate, like Tamogotchis and robotic "pets". Social interface robots use human-like social cues and facial expressions to convey messages often as tour guides or receptionists. These robots' behaviors are usually predetermined or reflexive. Socially receptive robots may appear similar to socially evocative robots, but socially receptive robots can additionally learn from their human interactions and update internal models. They are passively social and respond to interactions but do not engage with humans for social aims. Lastly, sociable robots have their own internal goals and motivations and engage in social interactions to benefit both themselves as well as others.

Many of the existing social robots tend to be either social interfaces or socially receptive,[194–197] or created with the primary purpose of exploring human-robot interactions[198–201] rather than social robot creation itself or socio-emotional modeling. Few existing robots take into account more complex socio-emotional models to enable a sociable robot with pro-social aims. Sonoh, et al. detail a robot with an emotional model based on the amygdala, the brain region responsible for social and emotional processing,[34] and executes the model on a field programmable gate array (FPGA) for fast processing.[202] Although it includes sensory recognition and classical conditioning,

it lacks any pro-social leanings.

5.10.2 Method

In this work we adapt the primate amygdala model previously detailed[144] for use in an interactive robotic application. The primate model, as well as our simplified version, not only takes in visual stimuli and processes it by assessing its emotional relevance, but also then determines an associated emotional state from which a physiological response could be mounted. Differing from existing work, these emotional states expand beyond purely fight or flight responses and include a pro-social response, all the while processed on neuromorphic hardware.

Our model focuses on the amygdala, which is believed to be the key part of the brain involved with the processing of social and emotional stimuli. The amygdala's primary role involves analyzing social interactions and contributing to stimulus appraisal, relevance detection, activation of neuroendocrine responses, and somatic motor expressions of emotions.[34] A diagram of the flow of information through the amygdala nuclei is shown in Figure 5.11. Though simplified, the amygdala model presented here still implements the processing of emotional stimuli, value appraisal, and provides outputs to mount an appropriate autonomic response.

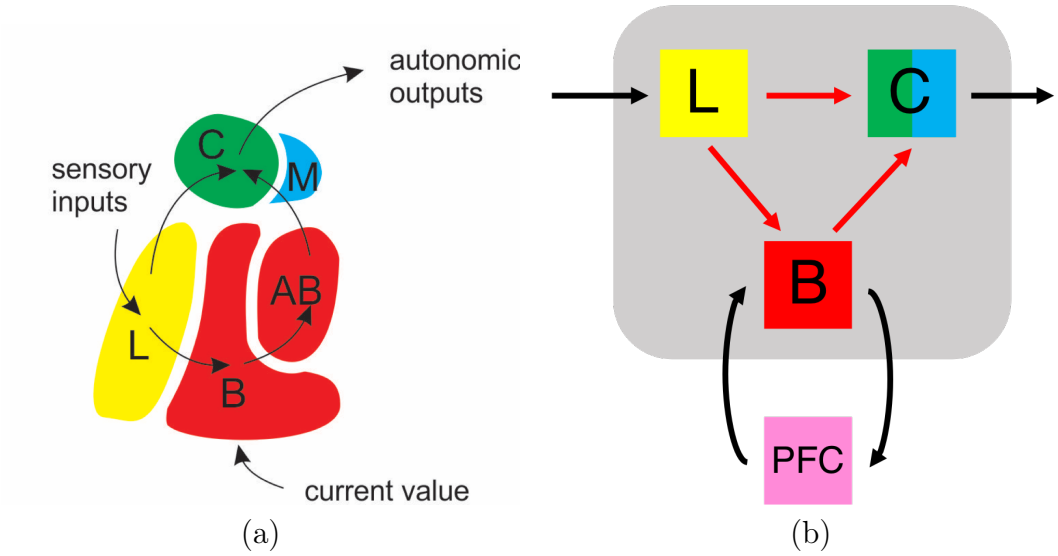


Figure 5.11: (a) Diagram of information processing within the amygdala. Information enters via the lateral nucleus (L), and then flows through the basal (B) nucleus and accessory basal (AB) nucleus where it is combined with a stored emotional state. Lastly information flows from the lateral and accessory basal nuclei to the central (C) and medial (M) nuclei, where it then leaves the amygdala via autonomic outputs. (b) Diagram of the amygdala model used in this robot. The model follows the same simplified processing flow, with the addition of a prefrontal cortex (PFC) which stores the previously determined emotional state. This image was adapted from the author's previous work.[144]

5.10.2.1 Model Overview

This model of the amygdala contains 2,000 neurons and consists of three main nuclei, a simplification and aggregation of general amygdala nuclei models. As seen in Figure 5.11, information first flows into the lateral nucleus, which uses its distributed spiking neurons to represent that same input. The lateral nucleus connects both to the basal and central nuclei. The connection from the lateral to basal nuclei deduces an emotional value from the input by translating the input into a two dimension vector, representing the valence and arousal of the input, each with one dimension of

the vector. This two dimensional emotional representation is adapted from existing work.[145] The basal nucleus also receives an input from another cortical region (pre-frontal cortex, PFC) which stores the current emotional state, thus allowing the new emotional value to be an average of the previous state and the new input. The basal nucleus connects to a central nucleus which then translates the valence and arousal into an emotional state. The central nucleus implements a winner-take-all configuration which outputs one of three states: pro-social, neutral, or distressed. Lastly, there is a connection from the lateral to central nucleus which only passes information if the lateral nucleus represents a distressing input, thus providing a fast pathway to the central nuclei to allow fast reactions to distressing stimuli. These states have been simplified from those implemented in the original primate amygdala model,[144] due to the simplified scenarios in which this robot operates.

5.10.3 Robot Structure and System Input and Output

This robot was constructed from a LEGO Mindstorms NXT robotics kit and can be seen in Figure 5.12. To provide visual information to the system, this robot uses Google's AIY Vision Kit.[203] The Vision Kit detects if the robot's social partner is smiling or frowning and returns a joy score (float between 0 and 1). This joy score is then passed as an input to the lateral nucleus model. The Vision Kit is secured with

LEGOs above the robot base.

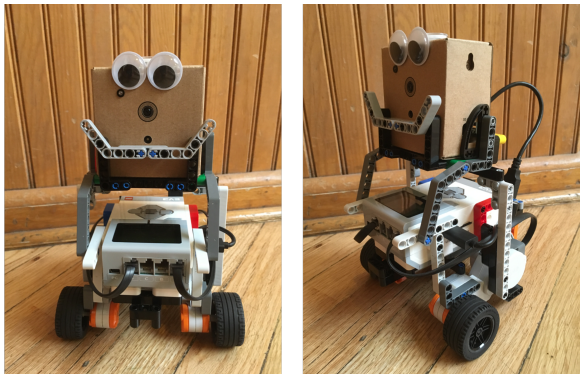


Figure 5.12: This robot was constructed from a LEGO Mindstorms NXT robotics kit. A Google AIY Vision Kit was mounted above the LEGO bumper car and processed the visual input prior to input to the model. All communication between the Vision Kit, model processing, and servo motors was performed wirelessly and described in Section 5.10.3.2. The googly eyes direct the robot's social partner's gaze towards the input sensor of the Vision Kit.

At the output of the model, the emotional state represented by the central nucleus is transformed into a one dimensional value, where an output of 1 indicates that the robot should drive towards the subject (the robot exhibits a state of "pro-social") and an output of -1 indicates that the robot should drive away from the subject (the robot exhibits a state of "distress"). This output value is sent from the central nucleus to the main Redis™ [‡] server, an open-source data structure store and message broker, which then sends commands to the robot's motor servos using ev3Link (see Section 5.10.3.2).

[‡]Redis is a trademark of Redis Labs Ltd

5.10.3.1 Neuromorphic Hardware

Neuromorphic hardware provides a path towards low-power, embedded processing, necessary for complex robots to interact with their environment in real time. Although the processing for this robot was not performed on the robot itself, a future version of this robot could have the associated chips mounted on its body for local, wired processing. Nonetheless, by using neuromorphic platforms for this robot implementation we are able to explore their tradeoffs and limitations within a real-time system. To implement the emotional model, three different neuromorphic processing systems were employed, including Intel's Loihi,[14] University of Manchester's SpiNNaker,[16,26,27] and Stanford University's Braindrop.[170]

By using the Nengo software package to create our model, we make use of Nengo's different "backends" to run parts of the model on different hardware. That is, we described the model in terms of the transformations that should be applied to the values represented by the neurons at different stages of processing. The backend software then determines the best way to achieve that mapping, given the constraints of the particular hardware at hand. This allowed us to design the parts of the model initially running on a standard CPU, and then transfer portions of the model to SpiNNaker, Loihi, and Braindrop, even though those different systems have extremely different neuron types. The software automatically finds the connection weights that will best approximate the desired functions given the available neural resources. The Nengo SpiNNaker backend is available at `https://github.`

`com/project-rig/nengo_spinnaker` and the Nengo Loihi backend is available at `https://github.com/nengo/nengo-loihi`.

5.10.3.2 Communication

Communication between the distributed nuclei of the model was achieved using Redis, an open-source data structure store and message broker. Each piece of hardware, including the AIY Vision Kit, LEGO Mindstorms NXT robot, and all neuromorphic chips, communicated through a central Redis server. The communication was implemented as non-blocking, allowing each hardware component to read and write at its own rate.

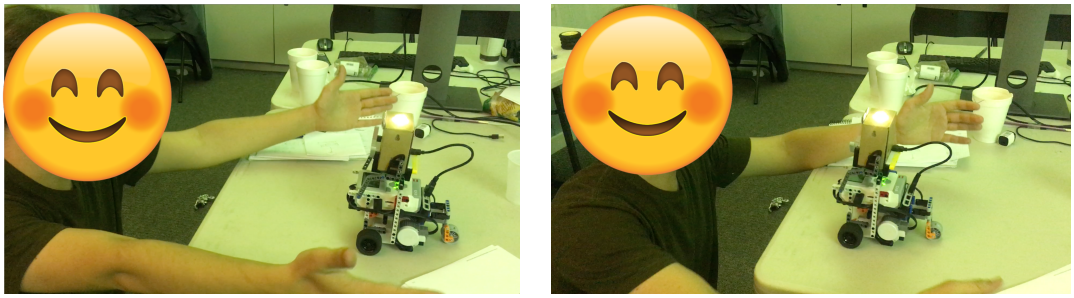
This non-blocking, distributed communication paradigm relied on the representation method employed by the NEF. In the NEF, the spiking pattern of a neural population encodes a particular value; by transmitting these values to the Redis server, rather than individual neuron spikes, each neuromorphic platform could process information at a different rate. This representation scheme enabled cross-platform computation by, for example, allowing Loihi's variable-length time step computation to coexist with Braindrop's analog real-time computation. This communication paradigm is particularly well-suited for human-computer interaction, where the rate of change of input (facial expression) is naturally slower than the neurons in the underlying model.

5.10.4 Results and Discussion

This robot was able to successfully interact in real time with persons of multiple ages, genders, and races. It accurately detected a social partner's facial expression, computing a pro-social emotional state in response to smiling, a distressed state in response to frowning, and a neutral state in response to neutral facial expression; the robot drove towards its social partner when pro-social, drove away when distressed, and remained stationary when neutral. Although the robot functioned well overall, it worked best in well-lit environments. Prior to implementation in the robot itself, the model was tested in simulation using Nengo and simulated input to verify its functionality.

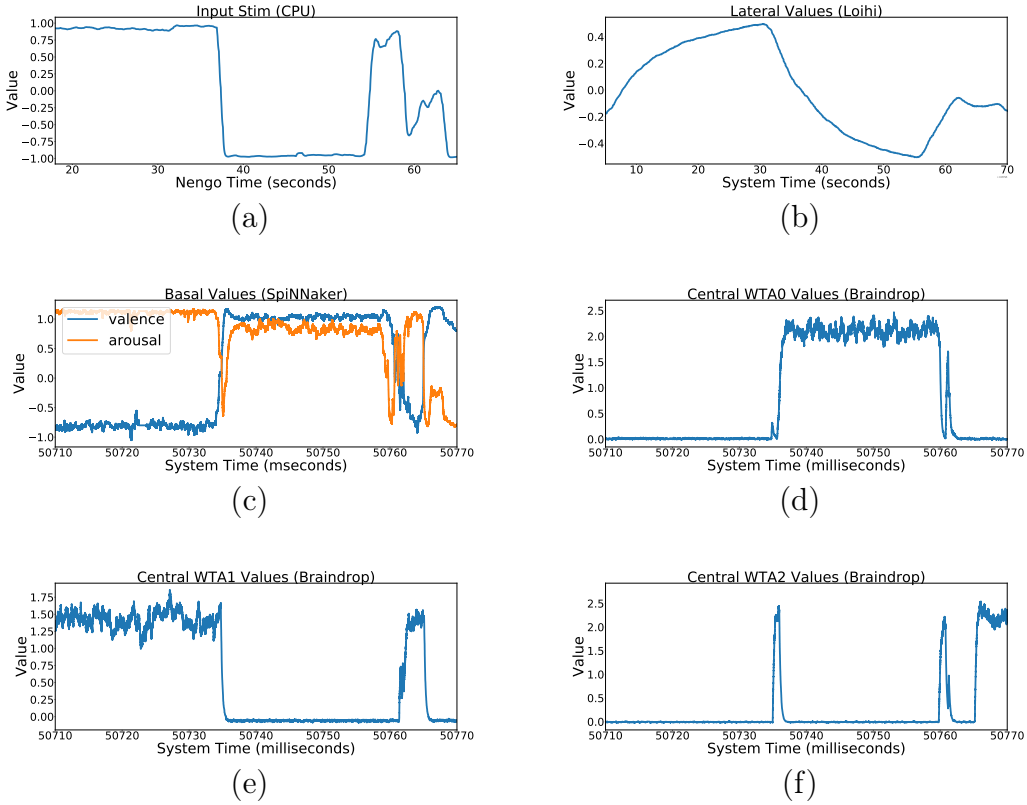
5.10.4.1 Spiking and Decoded Output

In addition to observing execution of the correct behavior, the neural output and the decoded values were recorded from the nuclei in real time. Figure 5.14 shows the represented or decoded output values of the input and of the nuclei as the social partner first smiles, then frowns, and then smiles again. Figure 5.15 shows the represented values and the associated neuron population spiking as the input changes in real-time.



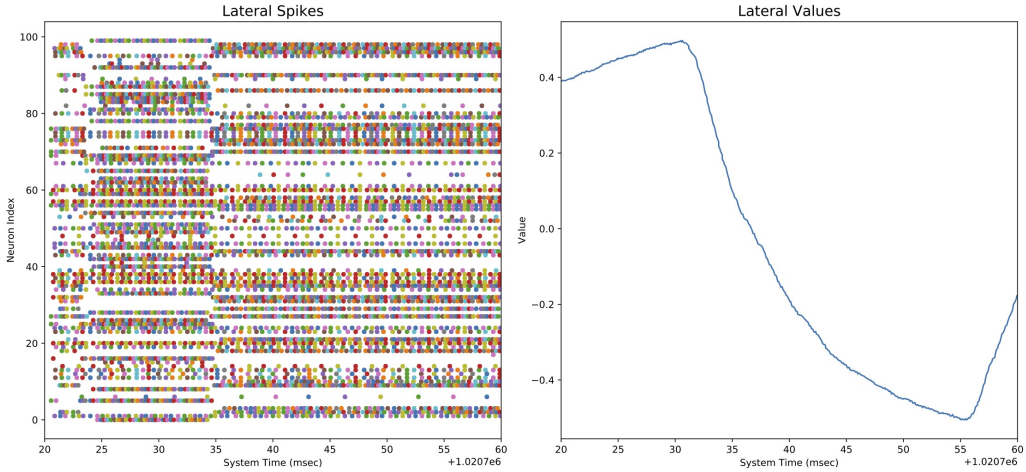
(a) (b)

Figure 5.13: Images illustrating the robot driving towards a smiling social partner after computing an internal emotional state of happiness.

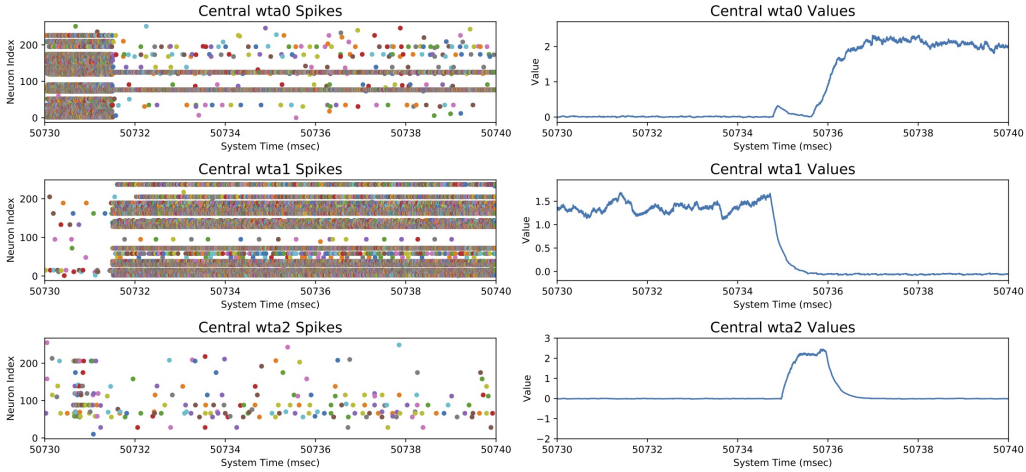


(a) (b) (c) (d) (e) (f)

Figure 5.14: Output values for (a) the input and (b) the lateral, (c) basal, and (d-f) central nuclei. Output values are decoded from each nucleus's distributed neural activity.



(a)



(b)

Figure 5.15: The represented value, and the associated neuron spikes for the lateral and central nucleus populations. This data was taken during the transition from an input of 1 (frown) to -1 (smile). The SpiNNaker implementation of Nengo does not support recording spikes in real-time; hence, only the lateral (Loihi) and central (Braindrop) nuclei are shown.

5.10.5 Conclusion

This work illustrates a proof-of-concept, socio-emotional robot that responds in real-time to smiling and frowning. It differs from many existing social robots because

it computes an internal emotional model based on socially-relevant visual inputs to determine the robot's current emotional state. Additionally, this robot utilizes a distributed processing system, processing each of the three nuclei of the model on a different neuromorphic chip, including SpiNNaker, Loihi, and Braindrop. This distributed processing illustrates advantages and limitations of these platforms, and provides a potential path forward for longer term, more computationally complex social robots and emotional modeling.

Bibliography

[1] R. Fischl, "On the best chebyshev approximation of an impulse response function at a finite set of equally-spaced points," Ph.D. dissertation, University of Michigan, 1966.

[2] D. S. Fischl, "Plasma-enhanced etching of tungsten, tungsten silicide, and molybdenum in chlorine containing discharges." Ph.D. dissertation, University of California, Berkeley, 1989.

[3] D. R. Hartree, "The eniac, an electronic computing machine," *Nature*, vol. 158, no. 4015, p. 500, 1946.

[4] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[5] S. B. Desai, S. R. Madhvapathy, A. B. Sachid, J. P. Llinas, Q. Wang, G. H. Ahn, G. Pitner, M. J. Kim, J. Bokor, C. Hu *et al.*, "Mos2 transistors with 1-nanometer gate lengths," *Science*, vol. 354, no. 6308, pp. 99–102, 2016.

BIBLIOGRAPHY

[6] S. I. Association, "2015 international technology roadmap for semiconductors (itrs)," Washington, DC, Tech. Rep., 2015.

[7] A. Szalay and J. Gray, "2020 computing: Science in an exponential world," *Nature*, vol. 440, no. 7083, p. 413, 2006.

[8] B. Marr. (2018) How much data do we create every day? the mind-blowing stats everyone should read. https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#3dae816460ba, Accessed 2019-04-08.

[9] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[10] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[11] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-cmos era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, vol. 45, pp. 4–26, 2013.

[12] S. Herculano-Houzel, "Scaling of brain metabolism with a fixed energy budget

per neuron: implications for neuronal activity, plasticity and evolution," *PloS one*, vol. 6, no. 3, p. e17514, 2011.

[13] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[14] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[15] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, 2019.

[16] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker System Architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[17] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen,

"Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[18] D. Brüderle, M. A. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M.-O. Schwartz *et al.*, "A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems," *Biological cybernetics*, vol. 104, no. 4-5, pp. 263–296, 2011.

[19] R. J. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings, "A multichip neuromorphic system for spike-based visual information processing," *Neural computation*, vol. 19, no. 9, pp. 2281–2300, 2007.

[20] S. Furber, "Large-scale neuromorphic computing systems," *Journal of neural engineering*, vol. 13, no. 5, p. 051001, 2016.

[21] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[22] J. Rivnay, H. Wang, L. Fenno, K. Deisseroth, and G. G. Malliaras, "Next-generation probes, particles, and proteins for neural interfacing," *Science Advances*, vol. 3, no. 6, p. e1601649, 2017.

[23] M. E. J. Obien, K. Deligkaris, T. Bullmann, D. J. Bakkum, and U. Frey, "Re-

vealing neuronal function through microelectrode array recordings," *Frontiers in neuroscience*, vol. 8, p. 423, 2015.

[24] A. C. Patil and N. V. Thakor, "Implantable neurotechnologies: a review of micro-and nanoelectrodes for neural recording," *Medical & biological engineering & computing*, vol. 54, no. 1, pp. 23–44, 2016.

[25] I. H. Stevenson and K. P. Kording, "How advances in neural recording affect data analysis," *Nature neuroscience*, vol. 14, no. 2, p. 139, 2011.

[26] M. M. Khan, D. R. Lester, L. A. Plana, A. Rast, X. Jin, E. Painkras, and S. B. Furber, "Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Ieee, 2008, pp. 2849–2856.

[27] S. Furber, F. Galluppi, S. Temple, and L. Plena, "The SpiNNaker Project," *Proceedings of the IEEE*, pp. 1–17, 2014.

[28] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[29] T. C. Stewart, "A technical overview of the neural engineering framework," tech. rep., Centre for Theoretical Neuroscience, Waterloo, ON, Tech. Rep., 2012.

[30] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing," *Preprint on ArXiv. http://arxiv. org/abs/1603.08270. Accessed*, vol. 27, Mar. 2016.

[31] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza, E. McQuinn, B. Shaw, N. Pass, and D. S. Modha, "Cognitive computing programming paradigm: A Corelet Language for composing networks of neurosynaptic cores," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–10.

[32] J. LeDoux, "The amygdala," *Current biology*, vol. 17, no. 20, pp. R868–R874, 2007.

[33] E. A. Murray, "The amygdala, reward and emotion," *Trends in cognitive sciences*, vol. 11, no. 11, pp. 489–497, 2007.

[34] D. Sander, J. Grafman, and T. Zalla, "The human amygdala: an evolved system for relevance detection," *Reviews in the Neurosciences*, vol. 14, no. 4, pp. 303–316, 2003.

[35] A. J. McDonald, "Cortical pathways to the mammalian amygdala," *Progress in neurobiology*, vol. 55, no. 3, pp. 257–332, 1998.

[36] A. Pitkänen and D. G. Amaral, "Demonstration of projections from the lateral nucleus to the basal nucleus of the amygdala: a pha-l study in the monkey," *Experimental brain research*, vol. 83, no. 3, pp. 465–470, 1991.

[37] H. Barbas, "Flow of information for emotions through temporal and orbitofrontal pathways," *Journal of anatomy*, vol. 211, no. 2, pp. 237–249, 2007.

[38] M. A. Belova, J. J. Paton, and C. D. Salzman, "Moment-to-moment tracking of state value in the amygdala," *Journal of Neuroscience*, vol. 28, no. 40, pp. 10 023–10 030, 2008.

[39] H. Ghashghaei, C. Hilgetag, and H. Barbas, "Sequence of information processing for emotions based on the anatomic dialogue between prefrontal cortex and amygdala," *Neuroimage*, vol. 34, no. 3, pp. 905–923, 2007.

[40] S. E. Morrison and C. D. Salzman, "Re-valuing the amygdala," *Current opinion in neurobiology*, vol. 20, no. 2, pp. 221–230, 2010.

[41] E. A. West, J. T. DesJardin, K. Gale, and L. Malkova, "Transient inactivation of orbitofrontal cortex blocks reinforcer devaluation in macaques," *Journal of Neuroscience*, vol. 31, no. 42, pp. 15 128–15 135, 2011.

[42] D. G. Amaral and J. Price, "Amygdalo-cortical projections in the monkey (macaca fascicularis)," *Journal of Comparative Neurology*, vol. 230, no. 4, pp. 465–496, 1984.

[43] Y. T. Cho, M. Ernst, and J. L. Fudge, "Cortico–amygdala–striatal circuits are organized as hierarchical subsystems through the primate amygdala," *Journal of Neuroscience*, vol. 33, no. 35, pp. 14 017–14 030, 2013.

[44] E. Bonda, "Organization of connections of the basal and accessory basal nuclei in the monkey amygdala," *European Journal of Neuroscience*, vol. 12, no. 6, pp. 1971–1992, 2000.

[45] J. Price and D. G. Amaral, "An autoradiographic study of the projections of the central nucleus of the monkey amygdala," *Journal of Neuroscience*, vol. 1, no. 11, pp. 1242–1259, 1981.

[46] J. E. LeDoux, "Evolution of human emotion: a view through fear," in *Progress in brain research*. Elsevier, 2012, vol. 195, pp. 431–442.

[47] L. W. Swanson, "The amygdala and its place in the cerebral hemisphere," *Annals of the New York Academy of Sciences*, vol. 985, no. 1, pp. 174–184, 2003.

[48] D. G. Amaral, M. D. Bauman, J. P. Capitanio, P. Lavenex, W. A. Mason, M. L. Mauldin-Jourdain, and S. P. Mendoza, "The amygdala: is it an essential component of the neural network for social cognition?" *Neuropsychologia*, vol. 41, no. 4, pp. 517–522, 2003.

[49] R. Adolphs, "What does the amygdala contribute to social cognition?" *Annals of the New York Academy of Sciences*, vol. 1191, no. 1, pp. 42–61, 2010.

[50] J. L. Freese and D. G. Amaral, *Neuroanatomy of the primate amygdala.* Guilford Press, 2009.

[51] C. M. Schumann and D. G. Amaral, "Stereological estimation of the number of neurons in the human amygdaloid complex," *Journal of Comparative Neurology*, vol. 491, no. 4, pp. 320–329, 2005.

[52] ——, "Stereological analysis of amygdala neuron number in autism," *Journal of Neuroscience*, vol. 26, no. 29, pp. 7674–7679, 2006.

[53] L. J. Chareyron, P. Banta Lavenex, D. G. Amaral, and P. Lavenex, "Stereological analysis of the rat and monkey amygdala," *Journal of Comparative Neurology*, vol. 519, no. 16, pp. 3218–3239, 2011.

[54] S. Herculano-Houzel, "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost," *Proceedings of the National Academy of Sciences*, vol. 109, no. Supplement 1, pp. 10 661–10 668, 2012.

[55] S. Herculano-Houzel, C. E. Collins, P. Wong, and J. H. Kaas, "Cellular scaling rules for primate brains," *Proceedings of the National Academy of Sciences*, vol. 104, no. 9, pp. 3562–3567, 2007.

[56] M. L. Phillips, W. C. Drevets, S. L. Rauch, and R. Lane, "Neurobiology of emotion perception i: The neural basis of normal emotion perception," *Biological psychiatry*, vol. 54, no. 5, pp. 504–514, 2003.

[57] C. P. Mosher, P. E. Zimmerman, and K. M. Gothard, "Response characteristics of basolateral and centromedial neurons in the primate amygdala," *Journal of Neuroscience*, vol. 30, no. 48, pp. 16 197–16 207, 2010.

[58] K. Braesicke, J. A. Parkinson, Y. Reekie, M.-S. Man, L. Hopewell, A. Pears, H. Crofts, C. R. Schnell, and A. C. Roberts, "Autonomic arousal in an appetitive context in primates: a behavioural and neural analysis," *European Journal of Neuroscience*, vol. 21, no. 6, pp. 1733–1740, 2005.

[59] S. Ballesta, C. P. Mosher, J. Szep, K. D. Fischl, and K. M. Gothard, "Social determinants of eyeblinks in adult male macaques," *Scientific reports*, vol. 6, p. 38686, 2016.

[60] L. Brothers, "The neural basis of primate social communication," *Motivation and emotion*, vol. 14, no. 2, pp. 81–91, 1990.

[61] A. Troisi, G. Schino, M. D'Antoni, N. Pandolfi, F. Aureli, and F. R. D'Amato, "Scratching as a behavioral index of anxiety in macaque mothers," *Behavioral and neural biology*, vol. 56, no. 3, pp. 307–313, 1991.

[62] C. M. Laine, K. M. Spitler, C. P. Mosher, and K. M. Gothard, "Behavioral triggers of skin conductance responses and their neural correlates in the primate amygdala," *Journal of neurophysiology*, vol. 101, no. 4, pp. 1749–1754, 2009.

[63] S. Grossberg, D. Bullock, and M. R. Dranias, "Neural dynamics underlying

impaired autonomic and conditioned responses following amygdala and orbitofrontal lesions." *Behavioral neuroscience*, vol. 122, no. 5, p. 1100, 2008.

[64] R. A. Hinde and T. Rowell, "Communication by postures and facial expressions in the rhesus monkey (macaca mulatta)," in *Proceedings of the Zoological Society of London*, vol. 138, no. 1. Wiley Online Library, 1962, pp. 1–21.

[65] K. M. Gothard, "The amygdalo-motor pathways and the control of facial expressions," *Frontiers in neuroscience*, vol. 8, p. 43, 2014.

[66] C. P. Mosher, P. E. Zimmerman, and K. M. Gothard, "Eye contact, a fundamental building block of social behavior, engages single unit activity in the monkey amygdala," *BMC neuroscience*, vol. 13, no. S1, p. P131, 2012.

[67] K. M. Gothard, C. P. Mosher, P. E. Zimmerman, P. T. Putnam, J. K. Morrow, and A. J. Fuglevand, "New perspectives on the neurophysiology of primate amygdala emerging from the study of naturalistic social behaviors," *Wiley Interdisciplinary Reviews: Cognitive Science*, 2017.

[68] C. P. Mosher, P. E. Zimmerman, and K. M. Gothard, "Videos of conspecifics elicit interactive looking patterns and facial expressions in monkeys." *Behavioral neuroscience*, vol. 125, no. 4, p. 639, 2011.

[69] A. Hall, "The origin and purposes of blinking," *The British journal of ophthalmology*, vol. 29, no. 9, p. 445, 1945.

[70] R. G. Linton, D. H. Curnow, and W. J. Riley, "The meibomian glands: an investigation into the secretion and some aspects of the physiology," *The British journal of ophthalmology*, vol. 45, no. 11, p. 718, 1961.

[71] D. R. Korb, D. F. Baron, J. P. Herman, V. M. Finnemore, J. M. Exford, J. L. Hermosa, C. D. Leahy, T. Glonek, and J. V. Greiner, "Tear film lipid layer thickness as a function of blinking." *Cornea*, vol. 13, no. 4, pp. 354–359, 1994.

[72] A. A. Cruz, D. M. Garcia, C. T. Pinto, and S. P. Cechetti, "Spontaneous eye-blink activity," *The ocular surface*, vol. 9, no. 1, pp. 29–41, 2011.

[73] M. K. Holland and G. Tarlow, "Blinking and thinking," *Perceptual and motor skills*, vol. 41, no. 2, pp. 403–406, 1975.

[74] A. R. Bentivoglio, S. B. Bressman, E. Cassetta, D. Carretta, P. Tonali, and A. Albanese, "Analysis of blink rate patterns in normal subjects," *Movement Disorders*, vol. 12, no. 6, pp. 1028–1034, 1997.

[75] D. A. Boehm-Davis, W. D. Gray, and M. J. Schoelles, "The eye blink as a physiological indicator of cognitive workload," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 44, no. 33. SAGE Publications Sage CA: Los Angeles, CA, 2000, pp. 6–116.

[76] Y.-F. Tsai, E. Viirre, C. Strychacz, B. Chase, and T.-P. Jung, "Task performance

and eye activity: predicting behavior relating to cognitive workload," *Aviation, space, and environmental medicine*, vol. 78, no. 5, pp. B176–B185, 2007.

[77] J. Oh, S.-Y. Jeong, and J. Jeong, "The timing and temporal patterns of eye blinking are dynamically modulated by attention," *Human movement science*, vol. 31, no. 6, pp. 1353–1365, 2012.

[78] F. Cummins, "Gaze and blinking in dyadic conversation: A study in coordinated behaviour among individuals," *Language and Cognitive Processes*, vol. 27, no. 10, pp. 1525–1549, 2012.

[79] S. Ballesta and J.-R. Duhamel, "Rudimentary empathy in macaques' social decision-making," *Proceedings of the National Academy of Sciences*, vol. 112, no. 50, pp. 15 516–15 521, 2015.

[80] A. Vrij, L. Akehurst, and S. Knight, "Police officers', social workers', teachers' and the general public's beliefs about deception in children, adolescents and adults," *Legal and Criminological Psychology*, vol. 11, no. 2, pp. 297–312, 2006.

[81] K. Takashima, Y. Omori, Y. Yoshimoto, Y. Itoh, Y. Kitamura, and F. Kishino, "Effects of avatar's blinking animation on person impressions," in *Proceedings of graphics interface 2008*. Canadian Information Processing Society, 2008, pp. 169–176.

[82] Y. Omori and Y. Miyata, "Estimates of impressions based on frequency of blink-

ing," *Social Behavior and Personality: an international journal*, vol. 29, no. 2, pp. 159–167, 2001.

[83] T. Nakano and S. Kitazawa, "Eyeblink entrainment at breakpoints of speech," *Experimental brain research*, vol. 205, no. 4, pp. 577–581, 2010.

[84] T. Nakano, N. Kato, and S. Kitazawa, "Lack of eyeblink entrainments in autism spectrum disorders," *Neuropsychologia*, vol. 49, no. 9, pp. 2784–2790, 2011.

[85] A. Mandel, S. Helokunnas, E. Pihko, and R. Hari, "Brain responds to another person's eye blinks in a natural settingâĂŤthe more empathetic the viewer the stronger the responses," *European Journal of Neuroscience*, vol. 42, no. 8, pp. 2508–2514, 2015.

[86] J. P. Capitanio, "Sociability and responses to video playbacks in adult male rhesus monkeys (macaca mulatta)," *Primates*, vol. 43, no. 3, pp. 169–177, 2002.

[87] C. P. Mosher, P. E. Zimmerman, and K. M. Gothard, "Neurons in the monkey amygdala detect eye contact during naturalistic social interactions," *Current Biology*, vol. 24, no. 20, pp. 2459–2464, 2014.

[88] P. Putnam, J. Roman, P. Zimmerman, and K. Gothard, "Oxytocin enhances gaze-following responses to videos of natural social behavior in adult male rhesus monkeys," *Psychoneuroendocrinology*, vol. 72, pp. 47–53, 2016.

[89] C. J. Machado, E. Bliss-Moreau, M. L. Platt, and D. G. Amaral, "Social

and nonsocial content differentially modulates visual attention and autonomic arousal in rhesus macaques," *PLoS One*, vol. 6, no. 10, p. e26598, 2011.

[90] F. VanderWerf, P. Brassinga, D. Reits, M. Aramideh, and B. Ongerboer de Visser, "Eyelid movements: behavioral studies of blinking in humans under different stimulus conditions," *Journal of neurophysiology*, vol. 89, no. 5, pp. 2784–2796, 2003.

[91] S. Shultz, A. Klin, and W. Jones, "Inhibition of eye blinking reveals subjective perceptions of stimulus salience," *Proceedings of the National Academy of Sciences*, vol. 108, no. 52, pp. 21 270–21 275, 2011.

[92] A. Vredeveldt, G. J. Hitch, and A. D. Baddeley, "Eyeclosure helps memory by reducing cognitive load and enhancing visualisation," *Memory & cognition*, vol. 39, no. 7, pp. 1253–1263, 2011.

[93] R. A. Nash, A. Nash, A. Morris, and S. L. Smith, "Does rapport-building boost the eyewitness eyeclosure effect in closed questioning?" *Legal and Criminological Psychology*, vol. 21, no. 2, pp. 305–318, 2016.

[94] T. Nakano, Y. Yamamoto, K. Kitajo, T. Takahashi, and S. Kitazawa, "Synchronization of spontaneous eyeblinks while viewing video stories," *Proceedings of the Royal Society B: Biological Sciences*, vol. 276, no. 1673, pp. 3635–3644, 2009.

[95] J. K. Hietanen, J. M. Leppänen, M. J. Peltola, K. Linna-aho, and H. J. Ruuhiala, "Seeing direct and averted gaze activates the approach–avoidance motivational brain systems," *Neuropsychologia*, vol. 46, no. 9, pp. 2423–2430, 2008.

[96] M. L. Spezio, P.-Y. S. Huang, F. Castelli, and R. Adolphs, "Amygdala damage impairs eye contact during conversations with real people," *Journal of Neuroscience*, vol. 27, no. 15, pp. 3994–3997, 2007.

[97] O. Dal Monte, M. Piva, J. A. Morris, and S. W. Chang, "Live interaction distinctively shapes social gaze dynamics in rhesus macaques," *Journal of neurophysiology*, vol. 116, no. 4, pp. 1626–1643, 2016.

[98] H. D. Freeman and S. D. Gosling, "Personality in nonhuman primates: a review and evaluation of past research," *American journal of primatology*, vol. 72, no. 8, pp. 653–671, 2010.

[99] A. Weiss, J. E. King, and L. Murray, *Personality and temperament in nonhuman primates*. Springer Science & Business Media, 2011.

[100] C. Neumann, M. Agil, A. Widdig, and A. Engelhardt, "Personality of wild male crested macaques (macaca nigra)," *PloS One*, vol. 8, no. 8, p. e69383, 2013.

[101] E. J. Feczko, E. Bliss-Moreau, H. Walum, J. R. Pruett Jr, and L. A. Parr, "The macaque social responsiveness scale (msrs): a rapid screening tool for assessing

variability in the social responsiveness of rhesus monkeys (macaca mulatta)," *PloS one*, vol. 11, no. 1, p. e0145956, 2016.

[102] S. Chevalier-Skolnikoff, "Facial expression of emotion in nonhuman primates," *Darwin and facial expression: A century of research in review*, pp. 11–89, 1973.

[103] H. Shimazaki and S. Shinomoto, "Kernel bandwidth optimization in spike rate estimation," *Journal of computational neuroscience*, vol. 29, no. 1-2, pp. 171–182, 2010.

[104] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[105] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects," *Journal of neurophysiology*, 2005.

[106] R. E. Kass, U. T. Eden, and E. N. Brown, *Analysis of neural data*. Springer, 2014, vol. 491.

[107] M. Warnecke, C. Chiu, J. Engelberg, and C. F. Moss, "Active listening in a bat cocktail party: adaptive echolocation and flight behaviors of big brown bats, eptesicus fuscus, foraging in a cluttered acoustic environment," *Brain, behavior and evolution*, vol. 86, no. 1, pp. 6–16, 2015.

[108] M. D. Gillette and H. F. Silverman, "A linear closed-form algorithm for source localization from time-differences of arrival," *IEEE Signal Processing Letters*, vol. 15, pp. 1–4, 2008.

[109] R. Adolphs, "Recognizing emotion from facial expressions: psychological and neurological mechanisms," *Behavioral and cognitive neuroscience reviews*, vol. 1, no. 1, pp. 21–62, 2002.

[110] K. M. Gothard, F. P. Battaglia, C. A. Erickson, K. M. Spitler, and D. G. Amaral, "Neural responses to facial expression and face identity in the monkey amygdala," *Journal of neurophysiology*, vol. 97, no. 2, pp. 1671–1683, 2007.

[111] R. Adolphs and D. Tranel, "Amygdala damage impairs emotion recognition from scenes only when they contain facial expressions," *Neuropsychologia*, vol. 41, no. 10, pp. 1281–1289, 2003.

[112] N. J. Emery and D. G. Amaral, "The role of the amygdala in primate social cognition," *Cognitive neuroscience of emotion*, pp. 156–191, 2000.

[113] K. C. Bickart, C. I. Wright, R. J. Dautoff, B. C. Dickerson, and L. F. Barrett, "Amygdala volume and social network size in humans," *Nature neuroscience*, vol. 14, no. 2, p. 163, 2011.

[114] U. Rutishauser, O. Tudusciuc, D. Neumann, A. N. Mamelak, A. C. Heller, I. B. Ross, L. Philpott, W. W. Sutherling, and R. Adolphs, "Single-unit responses

selective for whole faces in the human amygdala," *Current Biology*, vol. 21, no. 19, pp. 1654–1660, 2011.

[115] R. Kawashima, M. Sugiura, T. Kato, A. Nakamura, K. Hatano, K. Ito, H. Fukuda, S. Kojima, and K. Nakamura, "The human amygdala plays an important role in gaze monitoring: A pet study," *Brain*, vol. 122, no. 4, pp. 779–783, 1999.

[116] S. Eifuku, W. C. De Souza, R. Tamura, H. Nishijo, and T. Ono, "Neuronal correlates of face identification in the monkey anterior temporal cortical areas," *Journal of neurophysiology*, vol. 91, no. 1, pp. 358–371, 2004.

[117] A. Beyeler, P. Namburi, G. F. Glober, C. Simonnet, G. G. Calhoon, G. F. Conyers, R. Luck, C. P. Wildes, and K. M. Tye, "Divergent routing of positive and negative information from the amygdala during memory retrieval," *Neuron*, vol. 90, no. 2, pp. 348–361, 2016.

[118] D. Paré, "Role of the basolateral amygdala in memory consolidation," *Progress in neurobiology*, vol. 70, no. 5, pp. 409–420, 2003.

[119] J. Lin, M. Spraragen, and M. Zyda, "Computational models of emotion and cognition," in *Advances in Cognitive Systems*. Citeseer, 2012.

[120] M. Tamietto and B. De Gelder, "Neural bases of the non-conscious perception of

emotional signals," *Nature Reviews Neuroscience*, vol. 11, no. 10, pp. 697–709, 2010.

[121] C. D. Salzman and S. Fusi, "Emotion, cognition, and mental state representation in amygdala and prefrontal cortex," *Annual review of neuroscience*, vol. 33, pp. 173–202, 2010.

[122] K. H. Tieu, A. L. Keidel, J. P. McGann, B. Faulkner, and T. H. Brown, "Perirhinal-amygdala circuit-level computational model of temporal encoding in fear conditioning," *Psychobiology*, vol. 27, no. 1, pp. 1–25, 1999.

[123] A. A. Moustafa, M. W. Gilbertson, S. P. Orr, M. M. Herzallah, R. J. Servatius, and C. E. Myers, "A model of amygdala–hippocampal–prefrontal interaction in fear conditioning and extinction in animals," *Brain and cognition*, vol. 81, no. 1, pp. 29–43, 2013.

[124] J. L. Armony, D. Servan-Schreiber, J. D. Cohen, and J. E. LeDoux, "Computational modeling of emotion: Explorations through the anatomy and physiology of fear conditioning," *Trends in Cognitive Sciences*, vol. 1, no. 1, pp. 28–34, 1997.

[125] J. Armony, "Computational models of emotion," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 3. IEEE, 2005, pp. 1598–1602.

[126] F. Mannella, S. Zappacosta, M. Mirolli, and G. Baldassarre, "A computational model of the amygdala nucleiâĂŹs role in second order conditioning," *From Animals to Animats 10*, pp. 321–330, 2008.

[127] J. M. Salotti, "Computational model for amygdala neural networks." in *ESANN*, 2008, pp. 403–408.

[128] J. MorÉn, Christian Balkenius, "Emotional learning: a computational model of the amygdala," *Cybernetics & Systems*, vol. 32, no. 6, pp. 611–636, 2001.

[129] W. Haubensak, P. S. Kunwar, H. Cai, S. Ciocchi, N. R. Wall, R. Ponnusamy, J. Biag, H.-W. Dong, K. Deisseroth, E. M. Callaway *et al.*, "Genetic dissection of an amygdala microcircuit that gates conditioned fear," *Nature*, vol. 468, no. 7321, p. 270, 2010.

[130] P. H. Janak and K. M. Tye, "From circuits to behaviour in the amygdala," *Nature*, vol. 517, no. 7534, p. 284, 2015.

[131] M. Davis, D. L. Walker, L. Miles, and C. Grillon, "Phasic vs sustained fear in rats and humans: role of the extended amygdala in fear vs anxiety," *Neuropsychopharmacology*, vol. 35, no. 1, p. 105, 2010.

[132] M. Gamer, B. Zurowski, and C. Büchel, "Different amygdala subregions mediate valence-related and attentional effects of oxytocin in humans," *Proceedings of the National Academy of Sciences*, vol. 107, no. 20, pp. 9400–9405, 2010.

[133] F. Han, J. Ding, and Y. Shi, "Expression of amygdala mineralocorticoid receptor and glucocorticoid receptor in the single-prolonged stress rats," *BMC neuroscience*, vol. 15, no. 1, p. 77, 2014.

[134] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, no. 4771, pp. 1416–1419, 1986.

[135] A. Hurzook, O. Trujillo, and C. Eliasmith, "Visual motion processing and perceptual decision making," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 35, no. 35, 2013.

[136] *A spiking neuron model of movement and pre-movement activity in M1*, Salt Lake City, UT, 02/2011 2011.

[137] R. Singh and C. Eliasmith, "Higher-dimensional neurons explain the tuning and dynamics of working memory cells," *Journal of Neuroscience*, vol. 26, pp. 3667–3678, 2006.

[138] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a python tool for building large-scale functional brain models," *Frontiers in neuroinformatics*, vol. 7, p. 48, 2014.

[139] B. Bobier, T. C. Stewart, and C. Eliasmith, *The attentional routing circuit:*

receptive field modulation through nonlinear dendritic interactions. Nature Publishing Group, 2011.

[140] J. Conklin and C. Eliasmith, "A controlled attractor network model of path integration in the rat," *Journal of computational neuroscience*, vol. 18, no. 2, pp. 183–203, 2005.

[141] T. C. Stewart, T. Bekolay, and C. Eliasmith, "Learning to select actions with spiking neurons in the basal ganglia," *Frontiers in neuroscience*, vol. 6, p. 2, 2012.

[142] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[143] C. Eliasmith, *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[144] K. D. Fischl, T. C. Stewart, K. M. Gothard, and A. G. Andreou, "Exploring nuclear connectivity and function within the primate amygdala using the neural engineering framework." Program No. 525.03. 2018 Neuroscience Meeting Planner. San Diego, CA: Society for Neuroscience, 2018. Online.

[145] C. E. Osgood, W. H. May, M. S. Miron, and M. S. Miron, *Cross-cultural universals of affective meaning*. University of Illinois Press, 1975, vol. 1.

[146] K. D. Fischl, A. G. Andreou, T. C. Stewart, and K. Fair, "Implementation of the neural engineering framework on the truenorth neurosynaptic system," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2018, pp. 1–4.

[147] K. D. Fischl, G. Tognetti, D. R. Mendat, G. Orchard, J. Rattray, C. Sapsanis, L. F. Campbell, L. Elphage, T. E. Niebur, A. Pasciaroni *et al.*, "Neuromorphic self-driving robot with retinomorphic vision and spike-based processing/closed-loop control," in *Information Sciences and Systems (CISS), 2017 51st Annual Conference on*. IEEE, 2017, pp. 1–6.

[148] K. Fischl, K. Fair, W.-Y. Tsai, J. Sampson, and A. Andreou, "Spike propagation path planning on ibm truenorth neurosynaptic system," *Electronics Letters*, vol. 53, no. 15, pp. 1023–1025, 2017.

[149] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12)*. IEEE Computer Society, Nov. 2012, pp. 1–11.

[150] S. Koul and T. K. Horiuchi, "Path planning by spike propagation," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*. IEEE, 2015, pp. 1–4.

[151] S. Koziol, S. Brink, and J. Hasler, "A Neuromorphic Approach to Path Planning

Using a Reconfigurable Neuron Array IC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2724–2737, Nov. 2013.

[152] J. L. Krichmar, "Path Planning using a Spiking Neuron Algorithm with Axonal Delays," in *Proceedings of 2016 Conference on Evolutionary Computation (CEC)*, Jul. 2016, pp. 1219–1226.

[153] A. S. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. V. Arthur, P. A. Merolla, P. Datta, M. G. Tallada, B. Taba, A. Andreopoulos, A. Amir, S. K. Esser, J. Kusnitz, R. Appuswamy, C. Haymes, B. Brezzo, R. Moussalli, R. Bellofatto, C. Baks, M. Mastro, K. Schleupen, C. E. Cox, K. Inoue, S. Millman, N. Imam, E. McQuinn, Y. Y. Nakamura, I. Vo, C. Guo, D. Nguyen, S. Lekuch, S. Asaad, D. Friedman, B. L. Jackson, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/watt with," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'14)*. IEEE Press, Nov. 2014, pp. 27–38.

[154] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of Silicon Brains in the Nano-CMOS Era: Spiking Neurons, Learning Synapses and Neural Architecture Optimization," *Neural Networks*, vol. 45, pp. 4–26, Jun. 2013.

[155] C. A. Mead, "Neuromorphic Electronic Systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[156] F. C. Morabito, A. G. Andreou, and E. Chicca, "Neuromorphic Engineering: From Neural Systems to Brain-Like Engineered Systems," *Neural Networks*, vol. 45, pp. 1–3, May 2013.

[157] T. Hwu, J. Isbell, N. Oros, and J. Krichmar, "A self-driving robot using deep convolutional neural networks on neuromorphic hardware," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 635–641.

[158] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2011.

[159] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 1st ed. The MIT Press, Jul. 2009.

[160] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[161] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7, pp. 436–444, May 2015.

[162] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End

to End Learning for Self-Driving Cars," *arXiv preprint arXiv:1604.07316*, Apr. 2016.

[163] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[164] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-Road Obstacle Avoidance Through End-to-End Learning," in *Advances in Neural Information Processing Systems 18 (NIPS-2005)*, 2005.

[165] C. Mead, "Analog vlsi and neutral systems," *NASA STI/Recon Technical Report A*, vol. 90, 1989.

[166] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[167] ——, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[168] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza *et al.*, "Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–10.

[169] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[170] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, 2019.

[171] T. Bekolay, "Nengo-ocl," https://github.com/nengo/nengo-ocl, 2018.

[172] R. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, "A compact neural core for digital implementation of the neural engineering framework," in *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE.* IEEE, 2014, pp. 548–551.

[173] A. Mundy, J. Knight, T. C. Stewart, and S. Furber, "An efficient spinnaker implementation of the neural engineering framework," in *Neural Networks (IJCNN), 2015 International Joint Conference on.* IEEE, 2015, pp. 1–8.

[174] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon neurons that compute," in *International conference on artificial neural networks.* Springer, 2012, pp. 121–128.

[175] S. Menon, S. Fok, A. Neckar, O. Khatib, and K. Boahen, "Controlling articulated robots in task-space with spiking silicon neurons," in *Biomedical Robotics*

and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on. IEEE, 2014, pp. 181–186.

[176] F. Corradi, C. Eliasmith, and G. Indiveri, "Mapping arbitrary mathematical functions and dynamical systems to neuromorphic vlsi circuits for spike-based neural computation," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on.* IEEE, 2014, pp. 269–272.

[177] A. R. Voelker, B. V. Benjamin, T. C. Stewart, K. Boahen, and C. Eliasmith, "Extending the neural engineering framework for nonideal silicon synapses," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on.* IEEE, 2017, pp. 1–4.

[178] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[179] A. S. Cassidy, P. A. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B. D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–10.

[180] D. R. Mendat, "Cognitive and Brain-inspired Processing Using Parallel Algo-

rithms and Heterogeneous Chip Multiprocessor Architectures," Ph.D. dissertation, Ph.D Dissertation, Johns Hopkins University, Oct. 2017.

[181] K. L. Fair, "A biologically plausible sparse approximation solver on neuromorphic hardware," Ph.D. dissertation, Georgia Institute of Technology, 2017.

[182] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[183] T. U. o. M. APT Advanced Processor Technologies Research Group. (2019) Spinnaker publications. http://apt.cs.manchester.ac.uk/projects/SpiNNaker/Publications/, Accessed 2019-05-11.

[184] J. Broekens, M. Heerink, H. Rosendal *et al.*, "Assistive social robots in elderly care: a review," *Gerontechnology*, vol. 8, no. 2, pp. 94–103, 2009.

[185] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 271–281, 2003.

[186] S. Sabanovic, M. P. Michalowski, and R. Simmons, "Robots in the wild: Observing human-robot social interaction outside the lab," in *Advanced Motion*

Control, 2006. 9th IEEE International Workshop on. IEEE, 2006, pp. 596–601.

[187] H. G. Okuno, K. Nakadai, and H. Kitano, "Social interaction of humanoid robot based on audio-visual tracking," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems.* Springer, 2002, pp. 725–735.

[188] K. L. Koay, D. S. Syrdal, M. Ashgari-Oskoei, M. L. Walters, and K. Dautenhahn, "Social roles and baseline proxemic preferences for a domestic service robot," *International Journal of Social Robotics*, vol. 6, no. 4, pp. 469–488, 2014.

[189] M. Fridin, "Storytelling by a kindergarten social assistive robot: A tool for constructive learning in preschool education," *Computers & education*, vol. 70, pp. 53–64, 2014.

[190] B. Scassellati, "How social robots will help us to diagnose, treat, and understand autism," in *Robotics research.* Springer, 2007, pp. 552–563.

[191] J.-J. Cabibihan, H. Javed, M. Ang, and S. M. Aljunied, "Why robots? a survey on the roles and benefits of social robots in the therapy of children with autism," *International journal of social robotics*, vol. 5, no. 4, pp. 593–618, 2013.

[192] K. D. Fischl, A. B. Cellon, S. C. Terrence, T. K. Horiuchi, and A. G. Andreou, "Socio-emotional robot with distributed multi-platform neuromorphic

processing," in *Information Sciences and Systems (CISS), 2019 53rd Annual Conference on.* IEEE, 2019, pp. 1–6.

[193] C. Breazeal, "Toward sociable robots," *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 167–175, 2003.

[194] J. Saldien, K. Goris, B. Vanderborght, J. Vanderfaeillie, and D. Lefeber, "Expressing emotions with the social robot probo," *International Journal of Social Robotics*, vol. 2, no. 4, pp. 377–389, 2010.

[195] M. A. Salichs, R. Barber, A. M. Khamis, M. Malfaz, J. F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and D. Garcia, "Maggie: A robotic platform for human-robot social interaction," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on.* IEEE, 2006, pp. 1–7.

[196] T. Kanda, D. F. Glas, M. Shiomi, H. Ishiguro, and N. Hagita, "Who will be the customer?: A social robot that anticipates people's behavior from their trajectories," in *Proceedings of the 10th international conference on Ubiquitous computing.* ACM, 2008, pp. 380–389.

[197] P. Persson, J. Laaksolahti, and P. Lonnqvist, "Understanding socially intelligent agents - a multilayered phenomenon," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 31, no. 5, pp. 349–360, 2001.

[198] S. Morishima, "Modeling of facial expression and emotion for human communication system," *Displays*, vol. 17, no. 1, pp. 15–25, 1996.

[199] J. Kędzierski, R. Muszyński, C. Zoll, A. Oleksy, and M. Frontkiewicz, "Emys - emotive head of a social robot," *International Journal of Social Robotics*, vol. 5, no. 2, pp. 237–249, 2013.

[200] T. Ogata and S. Sugano, "Emotional communication between humans and the autonomous robot which has the emotion model," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 4. IEEE, 1999, pp. 3177–3182.

[201] M. Scheeff, J. Pinto, K. Rahardja, S. Snibbe, and R. Tow, "Experiences with sparky, a social robot," in *Socially Intelligent Agents*. Springer, 2002, pp. 173–180.

[202] S. Sonoh, S. Aou, K. Horio, H. Tamukoh, T. Koga, N. Shimo, and T. Yamakawa, "An implementation of intrinsic emotions on an autonomous robot with the emotional expression model of amygdala," in *SCIS & ISIS SCIS & ISIS 2008*. Japan Society for Fuzzy Theory and Intelligent Informatics, 2008, pp. 1894–1899.

[203] Google. (2018) Aiy. http://java.sun.com/products/jlf/ed1/dg/higq.htm, Accessed 2018-09-21.

Vita

Kate D. Fischl received a B. S. E. degree in Electrical Engineering from Princeton University in 2011, with certificates in Robotics and Intelligent Systems and Engineering-Biology. From 2011 until 2014 she worked at MIT Lincoln Laboratory in the Bioengineering and Systems Technologies Group. In 2014 she enrolled in the Electrical and Computer Engineering Ph.D. program at Johns Hopkins University. She received a National Science Foundation Graduate Research Fellowship in 2016 to fund her research. Her research focuses on neuromorphic modeling to better understand how the brain processes social and emotional stimuli, as well as understanding the gains and limitations of using neuromorphic hardware to execute such models.