Charles University
Faculty of Science
Study programme: Bioinformatics

Bc. David Beinhauer

Modeling spatio-temporal dynamics in primary visual cortex using deep neural network model

Modelování časoprostorové dynamiky v primární zrakové kůře pomocí modelu hluboké neuronové sítě

MASTER THESIS

Supervisor: Mgr. Ján Antolík, Ph.D.

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act. During the writing of this thesis, I utilized artificial intelligence (AI) tools in the following manner: I used ChatGPT-4o on my originally created text to improve its grammatical correctness, clarity, and readability.

In ..... date ....

Author's signature

I would like to express my sincere gratitude to my supervisor, Mgr. Ján Antolík, Ph.D., for his unwavering guidance and support throughout this project. I am also deeply grateful to Mgr. Luca Baroni for his valuable insights and constructive feedback. Finally, I extend my heartfelt thanks to all my friends, family members, and colleagues for their continuous support and encouragement throughout my studies.

Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Title: Modeling spatio-temporal dynamics in primary visual cortex using deep neural network model

Author: Bc. David Beinhauer

Department: Department of Cell Biology

Supervisor: Mgr. Ján Antolík, Ph.D.,

Abstract: Recent advances in computational neuroscience and machine learning have enabled increasingly sophisticated models of neuronal systems. However, standard deep neural networks (DNNs) often prioritize task performance over biological plausibility, limiting their ability to capture complex neural dynamics.

In this thesis, we propose a novel approach that integrates biologically inspired constraints into recurrent neural network (RNN) architectures to model the primary visual cortex (V1). Using synthetic data generated from a biologically detailed spiking neural network (SNN) model of cat V1, we develop RNN architectures incorporating anatomical structure alignment, excitatory-inhibitory neuron differentiation, biologically motivated neuronal modules, and synaptic depression mechanisms.

Our results show that RNN architectures without complex internal modules can predict mean neuronal responses but struggle to capture full system dynamics. Introducing shared DNN neuron modules improves dynamics slightly, while RNN-based neuron modules substantially enhance the temporal fidelity of predictions. Conversely, synaptic depression modules did not improve performance, likely due to computational constraints and suboptimal hyperparameter tuning.

This work demonstrates the promise of combining deep learning with biological realism to bridge the gap between predictive accuracy and interpretability, laying the groundwork for future applications to real neural recordings.

Keywords: visual cortex, recurrent networks, model deep neural networks

Contents

Introduction **3**

1 Early Visual System **7**
1.1 Neuron 7
1.1.1 Axon Overview 7
1.1.2 Action Potential 8
1.1.3 Synaptic Transmission 9
1.2 General Structure of the Early Visual System 9
1.3 Eye 12
1.3.1 Retina 12
1.4 Lateral Geniculate Nucleus 15
1.5 Primary Visual Cortex 16
1.5.1 Receptive Field Properties of V1 Cells 17
1.6 Extrastriate Visual Cortex 19

2 Computational Neuroscience **21**
2.1 Introduction to System Identification in the Visual System . . . 22
2.1.1 System Identification 22
2.1.2 Stimulus 23
2.1.3 Spike Trains 23
2.1.4 Mapping Function 24
2.2 Modeling Approaches 26
2.2.1 Classical Models 26
2.2.2 Deep Neural Network Approach 27
2.2.3 Spiking Neural Networks 29
2.3 Evaluation Methods 31
2.3.1 Pearson's Correlation Coefficient 32
2.3.2 Normalized Cross Correlation 33

3 Methods **35**
3.1 Spiking Model of Cat Primary Visual Cortex 35
3.1.1 Spiking Model Description 35
3.2 Artificial Dataset 37
3.2.1 Stimulus Sequence Structure 37
3.2.2 Experiment Definition 37
3.2.3 Dataset Preprocessing 39
3.2.4 Dataset Description 39
3.3 Model Description 41
3.3.1 Base Model Architecture 43
3.3.2 Additional Modules 48

4 Results **55**
4.1 Experimental Setup and Technicalities 55
4.2 Dataset Overview 56
4.2.1 Time Bin Merging Analysis 57
4.2.2 Model Subset Selection Analysis 66
4.3 Model Evaluation 69
4.3.1 Comparative Evaluation of Model Types 71
4.3.2 Temporal Dynamics of Predicted vs. Actual Neural Activity 73
4.3.3 Statistical Comparison of Model Performance 82
4.4 Analysis of Model Limitations and Improvement Opportunities 84
4.4.1 Free vs Teacher-Forced Prediction Analysis 84
4.4.2 Hyperparameter Grid Search 88
4.4.3 Impact of Training Dataset Size on Model Performance . 93

Conclusion **95**

Bibliography **97**

A Model Implementation and Evaluation Tools **109**

Introduction

Significant advances in neurobiology have been made in recent decades. The development of new technologies and methods has provided researchers with a diverse set of tools to study the brain. With the rise of highly parallelized computing, computational neuroscience has become one of the most important approaches to studying neuronal systems (Trappenberg [1]), offering a new perspective on brain function. It has enabled the simulation of large-scale neuronal networks, allowing us to analyze their behavior without relying solely on real-world experimental data subject to various limitations. As a result, we can now investigate brain systems in greater detail and gain a more precise understanding of their underlying principles.

With the rapid expansion of machine learning, particularly deep neural network (DNN) models, neuroscientists have also sought to apply these techniques. State-of-the-art convolutional DNN models have demonstrated outstanding performance in tasks such as image classification and object detection (Krizhevsky, Sutskever, and Hinton [2], Li et al. [3]). These methods have also been used to model certain regions of the brain, often yielding promising results. However, they come with significant limitations (Celeghin et al. [4]). Traditionally, researchers using DNNs typically disregard the anatomical structure and constraints of real neuronal networks. For instance, they usually rely exclusively on feed-forward architecture, whereas real brain networks are highly recurrent. As a consequence these DNN models typically only predict the average response of neurons over a period of time and hence do not fully capture the complex non-linear dynamics of the biological neural networks. Ultimately, currently, ML approaches in neuroscience prioritize task performance over biological plausibility and, as a consequence, interpretability.

On the other hand, the usage of biologically plausible models such as spiking neural network (SNN) models has been a fundamental approach in computational neuroscience (Ghosh-Dastidar and Adeli [5], Yamazaki et al. [6]). These models attempt to bridge biological knowledge with computational methods by incorporating biologically relevant constraints, thus mechanistically explaining how the computations performed by the brain are implemented in the biological neural

substrate. However, such SNN approaches face their own challenges, particularly limited ability to fit such models to data directly, therefore requiring a prior understanding of the system under study to generate precise mathematical formulations that define the behavior of the spiking network, limiting their flexibility and scalability (Izhikevich [7]).

One of the most studied, yet complex brain regions is the visual cortex. Thanks to extensive experimental research, the cortical subregion, the first to process visual information that arrives from the retina, called the primary visual cortex (V1), is the best understood (Miikkulainen et al. [8]). Extensive work on V1 involving both CNNs or SNNs have been undertaken, but both approaches have notable drawbacks (Niell and Scanziani [9]).

In this work, our objective is to overcome the limitations of both the CNN and SNN approaches by integrating biological constraints into the design of recurrent neural network (RNN) models to better understand the primary visual cortex (V1). In this proof-of-concept thesis, we will rely on synthetic data generated by a SNN model of cat V1 developed by Antolík et al. [10], which we will use to train our novel RNN architectures. Specifically, we focus on predicting synthetic neuronal responses in layer IV and layers II and III of V1, the first two visual cortical processing stages, using input from lateral geniculate nucleus (LGN) neurons.

While in this proof-of-concept initial study we train our model on synthetic SNN data, our long-term goal is to achieve strong predictive performance on real V1 neuronal recordings as well. Our novel V1 modeling RNN architecture incorporates the following biological constraints overlooked in previous ML models of visual system:

Anatomical structure alignment: The architecture of our model is layered according to known anatomical constraints. Each neuron in the network corresponds to a specific neuron in the reference SNN system. Such one-to-one correspondence between the reference SNN and RNN system facilitates straightforward principled validation of the ability of the novel RNN architectures to approximate the full dynamical complexity of the SNN systems. In future application to real data, such one-to-one correspondence would be dropped with the observable biological neurons linked to only small subset of the full RNN network.

Excitatory and inhibitory neuron differentiation: We explicitly distinguish between excitatory and inhibitory neurons, enforcing biologically plausible behavior within the architecture and ensuring that specific neuronal types function as expected.

Biologically inspired activation functions: Instead of standard activation functions such as ReLU or tanh, we introduce small DNN or RNN modules

in place of each reference SNN neurons, that approximate the biological transfer function of the given neuron. The parameters of these transfer modules are shared across each given anatomically defined layer. These modules aim to approximate the complexity of biological neuronal transfer function that involves number of processes that cannot be accounted for by simple monotonic point non-linearities, such as adaptation. They allow us to capture these non-linearities and allow neurons to retain a form of memory that adapts to previous stimuli.

Synaptic adaptation modules: To model the plasticity of neuronal synapses, we incorporate modules that adjust synaptic responses based on increased stimuli rates from specific neuronal layers. This mechanism aims to reflect the synaptic adaptation processes observed in biological synapses.

By embedding these anatomical constraints into our model, we aim to enhance both the predictive power and the interpretability of neural simulations, ultimately contributing to a better understanding of the selected brain region.

In this thesis, we have obtained the following results.

1. Novel biologically constrained RNN architectures without DNN neuron modules can be trained to accurately capture the mean neuronal responses; however, their ability to reconstruct the full SNN dynamics remains limited.

2. Incorporating shared DNN modules for neurons slightly improved the model's ability to capture dynamic behavior. Nevertheless, predictions largely remained dominated by the mean neuronal responses rather than the full system dynamics.

3. Differentiating inputs from excitatory and inhibitory neurons to the neuronal module does not appear to significantly influence model performance.

4. The use of RNN-based neuron modules substantially improved model performance in terms of capturing the temporal dynamics of neuronal responses and consistently outperformed all other tested model variants.

5. The inclusion of the synaptic depression module did not improve model performance and, surprisingly, appeared to worsen the model dynamics. We primarily attribute this outcome to limited computational resources and suboptimal model parameter settings, leading to underfitting and reduced overall performance compared to its potential capabilities.

The thesis is structured into several sections. In the initial chapters, we introduce the theoretical background of neurobiology and computational neuroscience,

with a focus on visual processing and, in particular, the primary visual cortex (V1). Alongside this, we review modeling approaches and machine learning methods relevant to our study. In the subsequent chapters, we describe in detail the architecture of our model, the dataset used, and the evaluation metrics employed. Finally, we present the experimental results of our study and provide an in-depth analysis of the findings.

Chapter 1

Early Visual System

In this thesis, we focus on modeling the primary visual cortex (V1), a region that plays a crucial role in early visual processing in the mammalian brain. V1 is among the most studied brain areas and has been a topic of extensive research for decades (Hubel and Wiesel [11]). This chapter highlights the aspects of the visual system that are most relevant to our work. For a more comprehensive overview, we recommend referring to the textbook Bear, Connors, and Paradiso [12] (3rd. Edition) and Goebel, Muckli, and Kim [13].

1.1 Neuron

The fundamental unit of neural processing is the *neuron*, a specialized cell composed of three primary parts: the soma, the axon, and the dendrites. The soma, or cell body, contains the nucleus and other organelles, and is structurally similar to other cell types. Dendrites are branched extensions that receive signals from other neurons. The axon is a long projection that transmits electrical impulses away from the soma toward other neurons.

1.1.1 Axon Overview

The axon originates from the *axon hillock* and terminates at the *axon terminal*, where it forms a *synapse* with the target cell. For efficient signal transmission, the axon contains specialized proteins for propagating electrical potentials. The axon terminal is rich in *synaptic vesicles* containing *neurotransmitters*—chemical messengers such as glutamate, gamma-aminobutyric acid (GABA), glycine, and acetylcholine that facilitate communication between neurons.

A detailed illustration of neuronal anatomy is provided in Figure 1.1. Further discussion on neuronal architecture can be found in Bear, Connors, and Paradiso

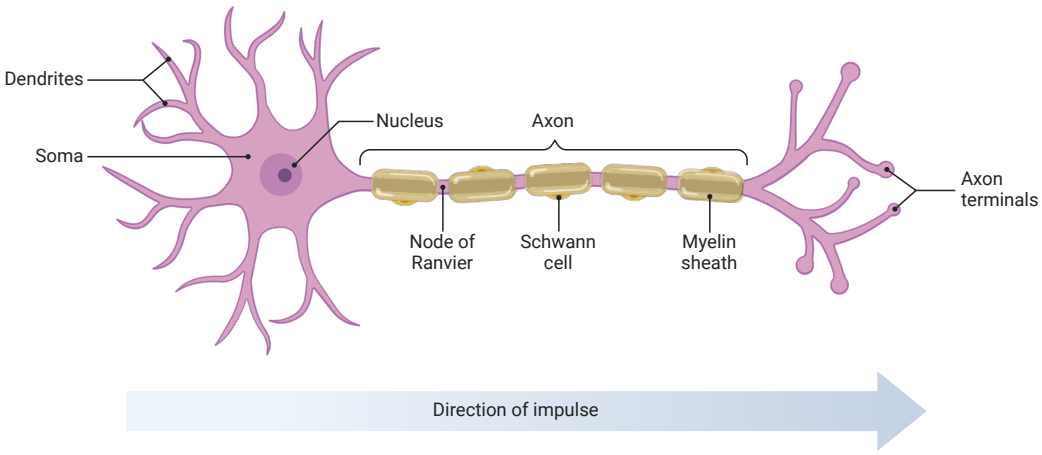

Figure 1.1 Neuron Anatomy. A schematic illustration of the structural components of a typical neuron. "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/m351rlh".

[12] (3rd Edition, p. 28–45).

1.1.2 Action Potential

The primary function of neurons is to propagate *action potentials*, enabling communication with other neurons or target cells. This process relies on differences in ion concentrations and electrical potentials across the cell membrane. Key concepts include the *membrane potential*—the voltage difference between the inside and outside of the neuron and *equilibrium potentials*, where electrochemical forces are balanced.

Ion types central to this mechanism include sodium (Na^+), potassium (K^+), chloride (Cl^-), and calcium (Ca^{2+}). All except potassium are typically found in higher concentrations outside the cell. Maintaining these gradients is energetically demanding and accounts for a substantial portion of the brain's energy consumption.

The resting membrane potential of a neuron is approximately −65 mV. Action potentials are initiated and propagated along the axonal membrane through the coordinated opening and closing of ion channels. The temporal profile of an action potential is shown in Figure 1.2.

To accelerate signal conduction over long distances, axons are often insulated by *myelin*, which enables *saltatory conduction* via gaps called *nodes of Ranvier*.

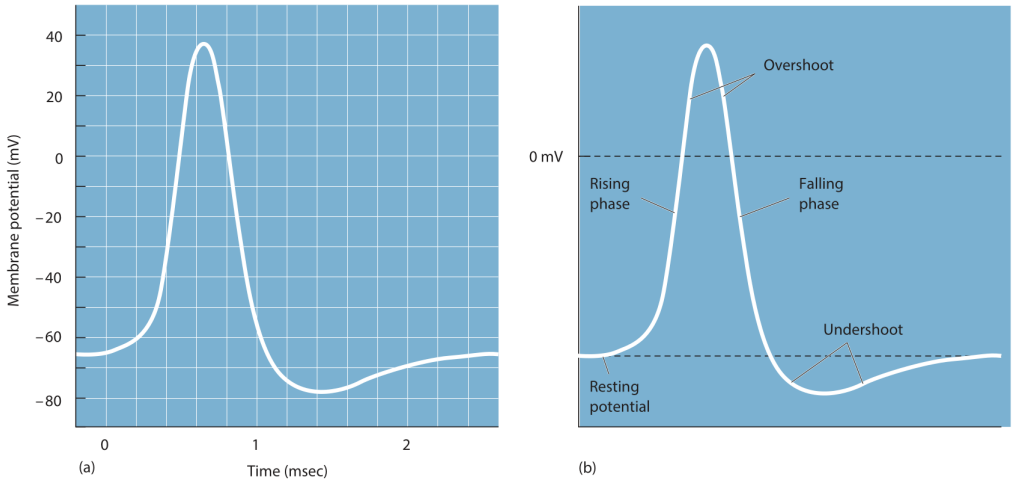

Figure 1.2 An action potential. a) Oscilloscope trace of an action potential. **b)** Annotated phases of an action potential. The figure and the labels are taken from *Neuroscience* (Bear, Connors, and Paradiso [12], 3rd Edition, p. 77).

This arrangement significantly increases propagation speed.

For a more detailed understanding of these processes, see Bear, Connors, and Paradiso [12], 3rd Edition, p. 52–98.

1.1.3 Synaptic Transmission

At the synapse, neurotransmitters are released by exocytosis when the action potential reaches the axon terminal. Synaptic connections can be either *excitatory*, leading to membrane depolarization and potential generation of an action potential, or *inhibitory*, which hyperpolarize the target cell and suppress firing.

Additionally, synaptic strength can adapt over short timescales based on recent activity, a phenomenon known as *synaptic depression* (Abbott et al. [14]). This adaptive behavior is commonly modeled in simulation frameworks and plays a critical role in temporal filtering and network dynamics.

For a more detailed understanding of these processes, see Bear, Connors, and Paradiso [12], 3rd Edition, p. 102–131.

1.2 General Structure of the Early Visual System

A significant part of the human brain is involved in visual processing. The initial stage of this process occurs within the so-called *early visual system*.

Visual information enters the system in the form of light, which passes through the eye's complex optical apparatus and is projected onto the retina. The retina

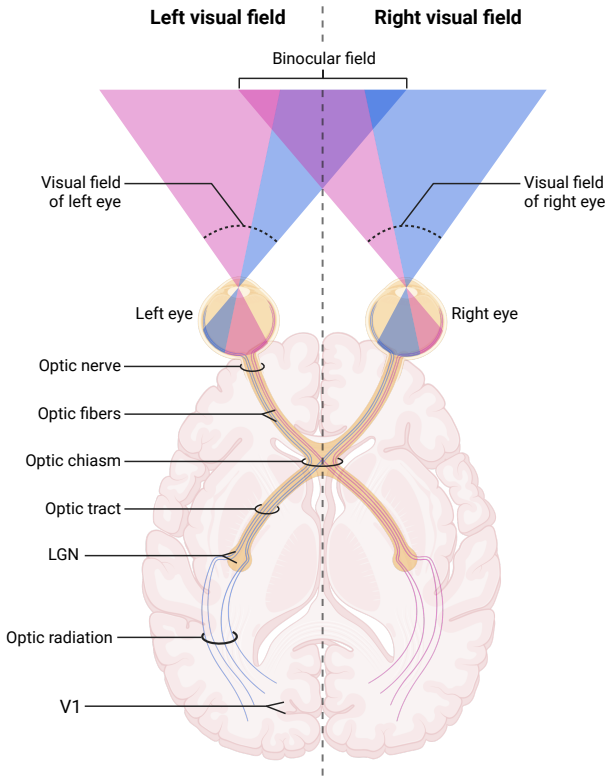

Figure 1.3 Anatomical Pathway of Visual Information Processing. Schematic representation of the anatomical pathway of visual information in the early visual system, tracing the route from the eye to the primary visual cortex. Adapted from Yaramothu and Alvarez [15] and elaborated with the description from Bear, Connors, and Paradiso [12] (3rd Edition, p. 311–314). "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/wpfx0a2".

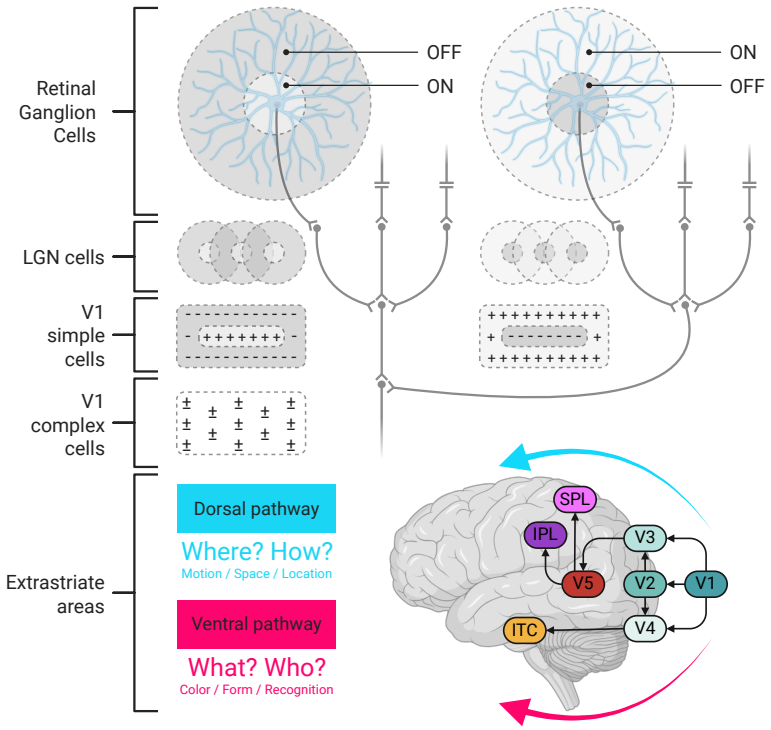

Figure 1.4 Stages of Visual Processing in the Brain. Schematic depiction of visual information flow from retinal ganglion cells to extrastriate cortical areas. The figure illustrates the receptive fields associated with each stage: starting with ON and OFF retinal ganglion cells and their concentric receptive fields, progressing through LGN cells, and into V1 where simple cells respond to centrally positioned oriented lines and complex cells to oriented lines regardless of exact position. The signal continues to higher-order extrastriate areas involved in advanced visual processing. Adapted from Felleman and Van Essen [16] and collaborated with the information from Bear, Connors, and Paradiso [12] (3rd Edition, p. 298–303, 316–337). "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/byl57wb".

is a thin layer of neural tissue responsible for detecting light and initiating signal processing. From there, signals travel through the *optic nerve* to the *optic chiasm*, where fibers are sorted based on the visual field, then continue via the *optic tract* to the *lateral geniculate nucleus* (LGN) of the thalamus.

The LGN performs early-stage processing and forwards most of the visual input to the *primary visual cortex* (V1) via the optic radiation. V1 is the first cortical region dedicated to visual analysis, where more complex computations begin. Signals leaving V1 proceed to higher-order visual areas responsible for advanced visual perception. The anatomical pathway from the eye to V1 is depicted in Figure 1.3, and the transformation of receptive fields throughout visual stages is illustrated in Figure 1.4.

1.3 Eye

One of the primary roles of the eye is to capture electromagnetic signals from the environment, focus them using its optical system, and project them onto the retina. The eye consists of several specialized structures that modulate and direct light. An anatomical overview is presented in Figure 1.5 and further discussed in Bear, Connors, and Paradiso [12] and Snell and Lemp [17].

1.3.1 Retina

A second essential function of the eye is the conversion and initial preprocessing of light into neural signals suitable for further brain processing. This task is carried out by the retina—a thin neural tissue layer located at the back of the eye. It is composed of multiple types of cells organized in distinct layers, each contributing to signal modulation. These include photoreceptors, bipolar cells, horizontal cells, amacrine cells, and ganglion cells.

The first cells involved in visual processing are the *photoreceptors*, namely *rods* and *cones*. These cells utilize photochemical reactions in their outer segments to convert light into chemical signals. Rods are highly sensitive to low light and detect a wide range of wavelengths, making them crucial for scotopic (low-light) vision. Cones, by contrast, are less light-sensitive but selective for specific wavelengths and are responsible for color vision and visual acuity.

Photoreceptors are distributed non-uniformly across the retina. The *fovea*, the central region of the retina, is densely packed with cones, while the peripheral retina predominantly contains rods. Additionally, the *optic disc*, or *blind spot*, is a region devoid of photoreceptors where the optic nerve exits the eye.

Upon activation, photoreceptors hyperpolarize and release the neurotransmitter glutamate. This signal is detected by *bipolar cells*, which serve as an

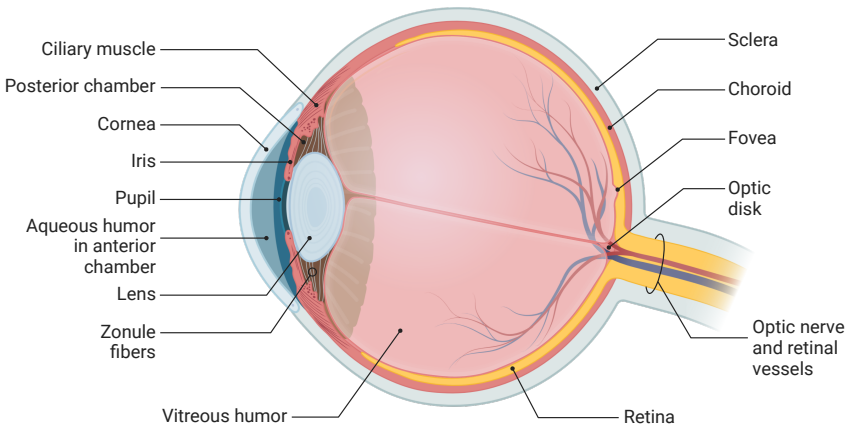

Figure 1.5 An anatomy of an eye. Illustration based on (Bear, Connors, and Paradiso [12], 3rd Edition, p. 283). "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/8n8epge".

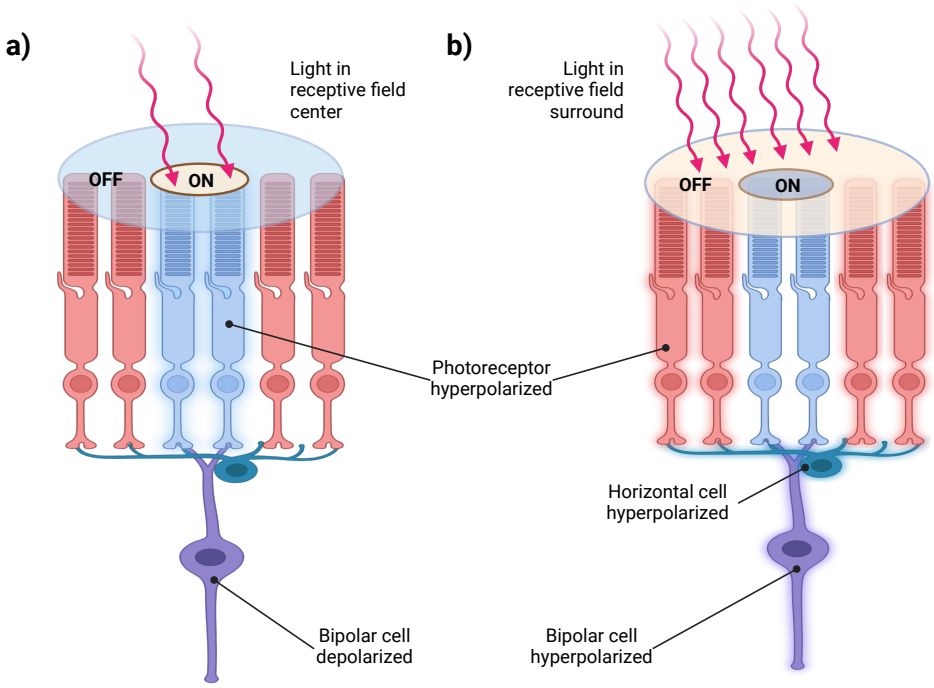

Figure 1.6 Direct and indirect pathways from photoreceptors to bipolar cells. **a)** An ON bipolar cell depolarizes in response to light in the center of its receptive field. **b)** The same cell hyperpolarizes in response to surround stimulation. Adapted from *Neuroscience* (Bear, Connors, and Paradiso [12], 3rd Edition, p. 300). "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/bpeqbrd".

intermediary step in the signal pathway. Notably, most retinal cells operate via graded potentials, with the exception of ganglion cells, which fire action potentials (Section 1.1.2).

The signal is further processed by *bipolar cells*, which aggregate input from varying numbers of photoreceptors depending on retinal location and cell type. This mechanism emphasizes foveal detail by focusing higher visual resolution centrally. Bipolar cells interact closely with *horizontal cells*, which contribute to lateral inhibition and temporal contrast.

Bipolar, ganglion, and LGN cells are categorized into *ON* and *OFF* types. ON bipolar cells are excited by illumination at the center of their receptive field, whereas OFF cells are excited when the center of receptive field is less illuminated than the surrounding area. This organization is visualized in Figure 1.6.

The signal is then transmitted to *ganglion cells*, which interact with *amacrine cells*. Signal processing at this level follows similar principles to those seen in bipolar and horizontal cell interactions. Ganglion cells are the first neurons in

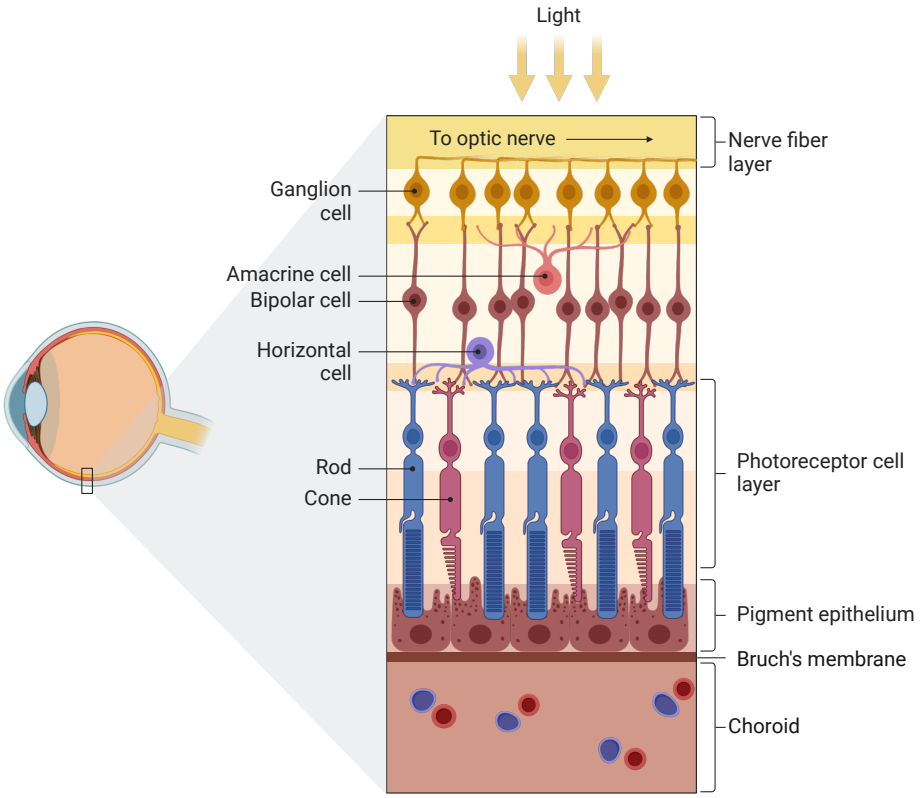

Figure 1.7 Layered anatomy of the retina. Adapted from Schwartz et al. [18], 4th Edition, p. 436 and Purves et al. [19], 6th Edition, p. 239. "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/oy5qz9v".

the pathway to generate action potentials, which is essential for transmitting signals over long distances to the brain. Their axons converge to form the optic nerve, terminating in the *lateral geniculate nucleus (LGN)* of the thalamus. The retina's layered structure is shown in Figure 1.7. For further details on retinal signal processing, see Bear, Connors, and Paradiso [12] (3rd Edition, p. 288–306).

1.4 Lateral Geniculate Nucleus

The next stage in visual processing occurs in the *lateral geniculate nucleus (LGN)*, located in the thalamus on each side of the brain. The LGN is a key structure responsible for the initial cortical-bound processing of visual signals. It serves as the primary termination site for most optic tract fibers (Bear, Connors, and

Paradiso [12], 3rd Edition, p. 313). From there, the majority of visual information is relayed to the *primary visual cortex (V1)* via the optic radiation. V1 is the principal cortical area involved in conscious visual perception (see Figure 1.3).

Damage to any part of the visual pathway—from the retina to the LGN to the V1—can result in various forms of visual impairment or blindness (Bear, Connors, and Paradiso [12], 3rd Edition, p. 313). Because of the structured, topographic nature of this pathway, lesions can often be traced to specific anatomical locations.

Interestingly, while the LGN receives direct input from the retina, the majority of its synaptic input actually originates from the cortex, particularly from V1. Although the exact functional role of this feedback remains under investigation, it is widely believed to support signal modulation and attentional feedback (Bear, Connors, and Paradiso [12], 3rd Edition, p. 318).

Anatomically, the LGN is composed of six distinct layers, each organized to separately process input from one eye. These layers exhibit concentric receptive fields similar to those found in retinal ganglion cells (see Figure 1.6). Importantly, the layers maintain *retinotopic* organization (Bear, Connors, and Paradiso [12], 3rd Edition, p. 319–320), preserving the spatial layout of the visual field across neurons.

Functionally, the LGN is responsible for maintaining multiple parallel processing streams, with distinct pathways specialized for detecting motion, color, and fine spatial details. These properties make the LGN a fundamental stage for visual signal refinement and dynamic modulation, and also mark the beginning of feedback-dependent processing. For a more comprehensive description of LGN function and structure, refer to Bear, Connors, and Paradiso [12] (3rd Edition, p. 315–320).

1.5 Primary Visual Cortex

The primary visual cortex (V1) is the first cortical area fully specialized for visual processing. It is located in the occipital lobe at the back of the brain. V1 is responsible for the initial cortical analysis of visual stimuli, including the encoding of orientation, size, motion, and color. Its outputs are split into two major processing streams: the parvocellular (P) stream, associated with form and color, and the magnocellular (M) stream, associated with motion and spatial information (Bear, Connors, and Paradiso [12], 3rd Edition, p. 330–332).

V1 is composed of six layers, conventionally labeled with Roman numerals (Bear, Connors, and Paradiso [12], 3rd Edition, p. 320). Input from the LGN is primarily directed to layer IV, which is further subdivided into layers IVA, IVB, IVC_{α}, and IVC_{β}. This layer primarily processes monocular input. The output is then relayed to layers II and III (Bear, Connors, and Paradiso [12], 3rd Edition,

p. 322), which are functionally similar and often grouped together as layer II/III. From this point onward, binocular integration becomes dominant. Outputs from these layers project to extrastriate areas such as V2 and V3, where higher-level visual processing occurs. Layers I, V, and VI are primarily involved in feedback and modulatory functions, with layers V and VI specifically contributing to recurrent processing from earlier visual stages (Bear, Connors, and Paradiso [12], 3rd Edition, p. 323).

Interlaminar connectivity is a key feature of V1, particularly within layer IV, where excitatory and inhibitory neurons form dense recurrent circuits that refine visual features. Inhibitory interneurons typically influence neurons within their own layers, while excitatory pyramidal neurons project across layers to support hierarchical processing (Bear, Connors, and Paradiso [12], 3rd Edition, p. 320–323).

V1 also exhibits *retinotopy* (mentioned in Section 1.4), a topographic organization whereby neighboring neurons correspond to adjacent locations in the retina, LGN, and visual field. This mapping is not uniform—central regions of the visual field are represented with greater cortical magnification than peripheral regions.

1.5.1 Receptive Field Properties of V1 Cells

Neurons in V1 are organized into overlapping, interconnected functional maps, each tuned to specific features of visual stimuli. These include:

Orientation selectivity: Starting in layer IVC_α and IVC_β, receptive fields become elongated, and neurons begin responding preferentially to stimuli of particular orientations. These preferences vary gradually across the cortical surface. Orientation selectivity gives rise to two classes of cells:

Simple cells: These neurons are selective for both the orientation and position of a stimulus. Their receptive fields contain distinct ON and OFF regions, and their responses can be modeled as the sum of aligned LGN inputs.

Complex cells: These neurons are orientation selective but respond across the entire receptive field, regardless of precise stimulus location. Their responses can be modeled as the pooled output of multiple simple cells.

Figure 1.4 illustrates the visual fields of both simple and complex cells. For a more detailed discussion, refer to Bear, Connors, and Paradiso [12] (3rd Edition, p. 325–329).

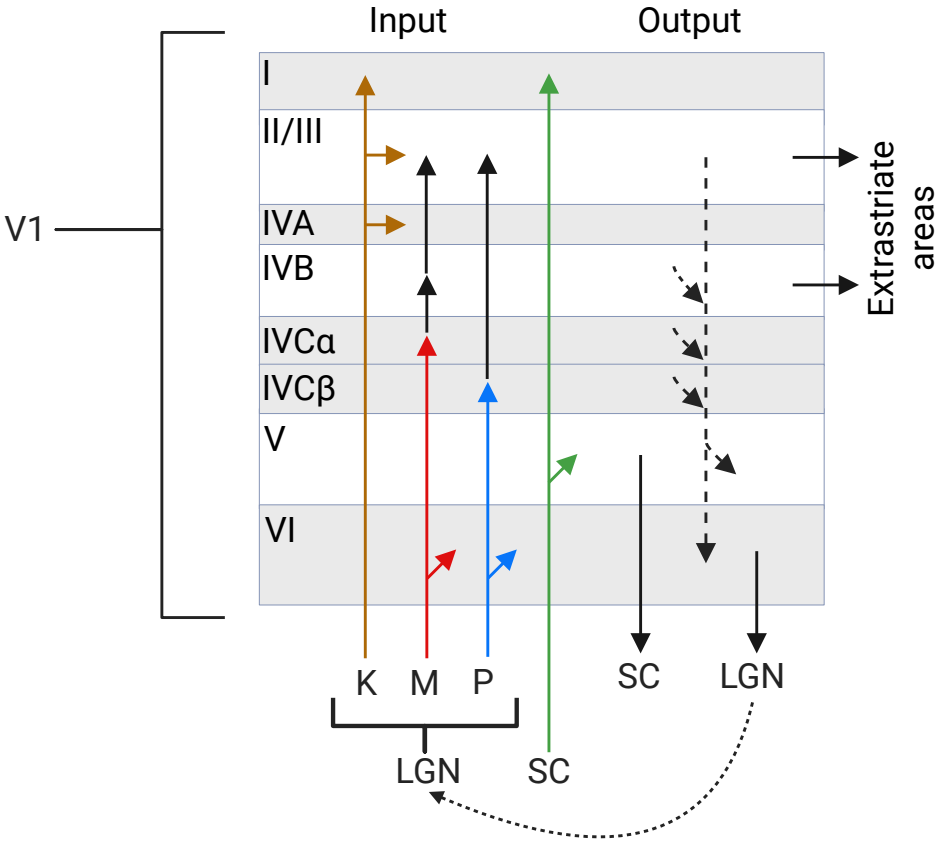

Figure 1.8 Layered Architecture of the Primary Visual Cortex (V1): This figure illustrates the laminar organization of V1. Different input streams are color-coded. Solid black lines represent feed-forward processing pathways originating from specific V1 layers, while dashed lines indicate feedback projections affecting all traversed layers, not just terminal targets. The symbols "K", "M", and "P" denote input streams from the lateral geniculate nucleus (LGN): koniocellular, magnocellular, and parvocellular pathways, respectively. "SC" refers to the superior colliculus. Adapted from Felleman and Van Essen [20] and Bear, Connors, and Paradiso [12] (3rd Edition, p. 318–324). "Created in BioRender. Beinhauer, D. (2025) https://BioRender.com/oo1k0jc".

Direction selectivity: Some V1 neurons respond preferentially to motion in a specific direction, typically orthogonal to the orientation axis.

Binocularity: While early layers (such as IV) are largely monocular, higher layers integrate signals from both eyes, allowing binocular perception and depth processing.

Other selectivities: Additional V1 neurons are tuned for motion sensitivity, depth cues, feedback modulation, and color processing.

For a more comprehensive description of V1 function and structure, refer to Bear, Connors, and Paradiso [12] (3rd Edition, p. 318–333).

The functional architecture and tuning properties of V1 lay the groundwork for computational modeling in visual neuroscience. By abstracting key principles such as orientation selectivity, hierarchical processing, and receptive field organization, researchers can build computational models that emulate early visual processing. These models not only shed light on underlying biological mechanisms but also provide a framework for testing how complex visual representations arise from structured cortical circuits.

1.6 Extrastriate Visual Cortex

Beyond V1, visual processing continues in a network of extrastriate cortical regions that extract increasingly abstract features of the visual scene. These areas include V2, V3, V4, and MT (middle temporal area) (Figure 1.4), each contributing to distinct aspects of perception such as object recognition, motion tracking, and depth analysis. Damage to these regions does not typically result in complete blindness, but instead leads to specific deficits, such as achromatopsia (loss of color perception) or motion blindness (inability to perceive fluid motion), depending on the affected region. For a more comprehensive description of extrastriate cortical regions function and structure, refer to Bear, Connors, and Paradiso [12] (3rd Edition, p. 333–337).

Chapter 2

Computational Neuroscience

Computational neuroscience is an interdisciplinary field that integrates mathematics, computer science, and neuroscience to enhance our understanding of the nervous system and the brain. With rapid advances in computational power and technological improvements in data acquisition, this field has gained increasing relevance. In particular, the recent rise of deep neural networks (DNNs) has significantly influenced computational neuroscience, offering powerful tools for modeling neural processes (Kietzmann, McClure, and Kriegeskorte [21]). These models are valued for their replicability and stability as they are not subject to biological noise and variability.

In computational neuroscience, models are typically categorized into three main types (Dayan and Abbott [22]). The first category consists of *descriptive* models, which summarize large sets of experimental data and characterize what the system does. Although these models may be biologically plausible, their primary aim is to describe rather than explain the system. An example of a descriptive model could be a mathematical representation of specific brain functions, such as the receptive field properties of neurons or neuronal spiking rates.

The second category includes *mechanistic* models, which seek to explain how a system functions at an anatomical and physiological level. These models incorporate biological structures and mechanisms to provide insight into neural dynamics and interactions.

The third category comprises *interpretive* models that employ computational principles to explore the cognitive and behavioral functions of the brain. These models address questions related to why the nervous system operates in a particular way, providing theoretical frameworks for understanding neural computation and cognition.

It is often challenging to assign a given model to a single category, as many models exhibit characteristics that span multiple types. This thesis serves as an example of such an overlap, as it aims not only to describe neural behavior but also

to enhance its interpretability and provide insights into underlying mechanisms. Consequently, the proposed model integrates aspects of all three categories to some extent.

In the following sections, we will introduce key concepts in computational neuroscience, discuss selected modeling approaches commonly used in the field, and outline typical evaluation methodologies.

2.1 Introduction to System Identification in the Visual System

Since this thesis focuses on the study of the visual system, we will primarily introduce concepts closely related to its analysis. In particular, the primary visual cortex 1.5 and the broader visual system are among the most extensively studied areas in computational neuroscience. As a result, many foundational terms and methodologies in the field have been defined using the visual system as a reference (Dayan and Abbott [22]).

Characterizing the relationship between stimulus and neuronal response is a complex task, as it depends not only on the stimulus itself but also on various factors such as the temporal state of the neuron, the interval between stimuli, and other contextual influences. Due to these factors, neuronal responses can vary between trials, even when the same stimulus is presented. This variability makes the precise prediction of individual neuronal spikes highly challenging. Instead, a more practical approach is to model the probability distribution of responses across trials and evaluate overall performance.

Typically, multiple neurons respond to a given stimulus, meaning that stimulus properties are encoded across a population of neurons rather than by individual units. Consequently, it is often more informative to analyze and accurately model the activity of neuronal populations rather than focusing solely on single-neuron predictions.

A comprehensive overview of system identification in neuroscience can be found in Butts [23], which serves as a key reference for the following sections.

2.1.1 System Identification

System identification aims to estimate and describe an unknown system based on measured data (Butts [23]). In the context of the visual system, the system of interest is the corresponding neuronal population, and the measured data typically consist of visual stimuli and their corresponding neuronal responses. A common approach in system identification is to separate the data into input

and output components, where the input represents the stimuli, and the output represents the neural responses.

Formally, system identification in this context can be expressed as follows:

$$r = f(s) \tag{2.1}$$

where s denotes the stimulus, r represents the neuronal response typically in the form of spikes or spike trains and f is the unknown mapping function that characterizes the system. The goal of system identification is to estimate the function f based on observed data.

2.1.2 Stimulus

A stimulus is any input signal capable of eliciting a response. It is typically represented as a vector, matrix, or sequence of such structures, depending on the context. In the visual system, the stimulus often takes the form of an image or a sequence of images (i.e., a video) represented as a matrix.

A fundamental challenge in neuroscience is determining the most appropriate stimuli for a given task (Carandini et al. [24]). Early research predominantly used simple artificial stimuli, such as bars or sinusoidal grating patterns, as demonstrated in Hubel and Wiesel [11]. However, these artificial stimuli lack ecological validity as they do not naturally occur in real-world environments.

A widely used alternative is *white-noise stimuli* (Dayan and Abbott [22] and Chichilnisky [25]), in which each pixel of an image stimulus is independently randomized. Properly designed, this approach ensures a uniform coverage of the entire space of possible visual stimuli. While white-noise stimuli have been shown to be informative, they also present drawbacks (Talebi and Baker [26]). Specifically, since neurons evolved to primarily process natural stimuli, white-noise stimuli may not effectively activate highly specialized neurons for certain aspects of natural vision.

As a result, many recent studies have shifted towards the use of *natural stimuli* (Sonkusare, Breakspear, and Guo [27], Lurz et al. [28], and Antolík et al. [10]). Natural stimuli allow researchers to investigate not only the general properties of the neuronal response but also the specific neuronal selectivity to features inherent in real-world visual environments.

2.1.3 Spike Trains

Neurons transmit information through action potentials 1.1.2, commonly referred to as *spikes*. Although spikes can vary in duration, amplitude, and shape, they are

typically treated as binary events. Given their brief duration (approximately 1 ms), spikes are often modeled as instantaneous occurrences. A sequence of spikes over time is termed *spike train*.

Since spike trains exhibit variability between trials, they are commonly analyzed using a statistical measure known as the *firing rate*. Following Dayan and Abbott [22], we define the firing rate as follows:

$$r = \frac{n}{T} = \int_0^T \rho(\tau)\,\mathrm{d}\tau \tag{2.2}$$

where r represents the firing rate, n is the number of spikes during the trial, T is the trial duration, and $\rho(\tau)$ is the spike density function in continuous time.

Furthermore, a *time-dependent firing rate* can be defined over discrete time intervals. If an experiment of duration T is divided into N equal time bins of duration $T_{\text{int}} = \frac{T}{N}$, the firing rate in the time bin t is given by the following.

$$r(t) = \frac{n(t)}{T_{\text{int}}} = \int_{(t-1)T_{\text{int}}}^{(t)T_{\text{int}}} \rho(\tau)\,\mathrm{d}\tau \tag{2.3}$$

However, this metric is typically averaged over multiple trials, as increasing temporal resolution also increases noise as a result of spiking variability. If the time bins are very small (e.g., 1 ms), the firing rate becomes a high-resolution measure of the spiking activity. Given that an individual spike lasts approximately 1 ms (Dayan and Abbott [22]), spike counts in such fine-grained bins can be treated as binary values within a single trial. Averaging these values across trials then yields the probability of a spike occurring at each time step. In practice, there is a trade-off between temporal resolution and noise: higher resolution captures finer temporal details, but introduces more variability, requiring more data for reliable estimates. Computational complexity is another factor, as finer resolutions demand increased processing power to mitigate noise. In this thesis, we prioritize a balance between these factors and use a 20 ms time bin for discretizing spike trains.

2.1.4 Mapping Function

The final component of the system identification problem is the mapping function f. In the study of neural systems, the most common approach to describe the behavior of the system is to use a mathematical representation (Butts [23]).

There are various approaches to modeling the mapping function. According to Butts [23], several key factors must be taken into account when selecting an appropriate model. These include: (1) the ease of parameter optimization, (2) the

model's ability to generalize to unseen data, and (3) the insights it can provide into the system's underlying properties.

These factors are heavily influenced by the experimental data used. Researchers must carefully consider the quantity, quality and type of stimulus data to best suit the chosen model, as briefly discussed in Section 2.1.2.

Maximum a Posteriori Estimation

In an ideal scenario, statistical models would be derived solely from experimental recordings. However, in practice, the task typically involves estimating the model parameters. Choosing the model's mathematical form and its parameters is often the most critical step in model design.

One of the most common approaches to parameter optimization in neuroscience is *maximum a posteriori estimation* (MAP) (Wu, David, and Gallant [29] and Butts [23]). Unlike classical maximum likelihood estimation (MLE) (Alpaydin [30]), MAP incorporates additional model constraints via prior distributions, which is especially valuable in neuroscience, where spiking neuronal data are often noisy. The use of prior knowledge about the system helps to improve the robustness of parameter estimation.

Let us denote the model parameters as Θ, and assume that we have M pairs of observed input-output data (s_i, r_i), where $s \in S$ represents the stimulus 2.1.2 and $r \in R$ is the corresponding neural response 2.1.3. These pairs are assumed to follow the relationship of the system defined in Equation 2.1.1. The MAP framework aims to maximize the posterior distribution of the model parameters:

$$
\begin{aligned}
\Theta_{MAP} &= \arg\max_{\Theta} p(\Theta|S,R) \\
&= \arg\max_{\Theta} p(R|\Theta,S)\cdot p(\Theta) \\
&= \arg\max_{\Theta} \left[\prod_{i=1}^{N} p(r_i|f_\Theta(s_i))\cdot p(\Theta)\right] \\
&= \arg\min_{\Theta} \left[\underbrace{-\sum_{i=1}^{M}\overbrace{\log p(r_i|f_\Theta(s_i))}^{\text{Likelihood}}}_{\substack{\text{Negative Log Likelihood}\\ \text{(Loss function)}}} - \underbrace{\overbrace{\log p(\Theta)}^{\text{Prior}}}_{\substack{\text{Negative-log prior}\\ \text{(Regulatizer)}}}\right]
\end{aligned}
\tag{2.4}
$$

This formulation is derived from basic probability rules. For a complete explanation, refer to Alpaydin [30], Wu, David, and Gallant [29], and Butts [23].

In the context of our task, p represents the noise distribution in the data, f_Θ is the mapping function (i.e., the model) with parameters Θ, and $p(\Theta)$ encodes our prior knowledge of the system. The inclusion of this prior term is what distinguishes MAP from MLE.

The primary reason for favoring MAP in neuroscience is that, in the absence of strong prior knowledge, the noise in the data can easily lead to overfitting, especially given the typically limited size of experimental datasets. Moreover, the likelihoods of different models may be very similar. In this formulation, the first term represents a loss function quantifying how well the model fits the data, while the second term acts as a regularizer penalizing model complexity. The MAP framework thus aims to find an optimal balance between data fitting and model simplicity.

2.2 Modeling Approaches

One of the most important aspects of computational neuroscience is the selection of an appropriate model. The choice of the model directly influences not only how accurately we can predict neural responses but also the extent to which we can extract meaningful insights into the underlying biological mechanisms. More complex models may better capture the behavior of the system, but they are also more prone to overfitting (as briefly discussed in Section 2.1.4).

In this section, we explore three major modeling approaches commonly used in computational neuroscience. We begin with the simplest models, which rely on straightforward mathematical methods and are easy to fit, referred to as *classical models*. Next, we consider models based on deep neural networks (DNNs) and, finally, we describe spiking neural networks (SNNs) which aim to reflect the biological properties of neurons.

The primary source for this section is Butts [23], which outlines the development of data-driven approaches in the study of the visual neural system. For readers seeking a deeper understanding of the methods described here, we recommend consulting the cited work.

2.2.1 Classical Models

In the early stages of computational neuroscience, the models were typically simple and aimed to describe neural behavior using formulations that were easy to optimize. The most basic example is *linear model*, where a neuron's response to multiple stimuli is represented as a weighted sum:

$$r_{\mathrm{L}} = f_{\mathrm{linear}}(\boldsymbol{s}) = \boldsymbol{w}\boldsymbol{s} = \sum_i w_i s_i$$

Here, r_{L} is the neuronal response, f_{linear} denotes the mapping function, $\boldsymbol{s}$ is the stimulus vector, and $\boldsymbol{w}$ is the vector of weights (often called a *filter*).

This model can be extended with non-linearity, resulting in the *linear-nonlinear model* (LN):

$$r_{\mathrm{LN}} = g_{\mathrm{LN}}(r_{\mathrm{L}}) = g_{\mathrm{LN}}(f_{\mathrm{linear}}(\boldsymbol{s}))$$

In this case, g_{LN} is a non-linear function, such as a sigmoid or exponential, which introduces thresholding or saturation effects, and r_{LN} is the neuron's final predicted response.

For many years, LN models were a cornerstone of sensory neuroscience. They provided intuitive interpretations of neuronal responses from both a computational perspective (Hubel and Wiesel [11] and Movshon, Thompson, and Tolhurst [31]) and a biological one (Mohanty, Scholl, and Priebe [32], Shapley [33], and Poirazi, Brannon, and Mel [34]). In particular, they performed well when modeling neurons in the early visual system (Shapley [33], Baccus and Meister [35], and Carandini et al. [24]).

Despite these advantages, classical models have limitations. They often fail to capture the complex nonlinearities and temporal dynamics observed in real neural systems. As a result, more advanced models have increasingly been adopted, offering deeper insight into the visual pathway and allowing for analysis at a larger scale (Maheswaranathan et al. [36], Butts et al. [37], and Keat et al. [38]).

2.2.2 Deep Neural Network Approach

While it has been proven that cascades of linear-nonlinear (LN) units 2.2.1 can approximate complex functions (Cybenko [39] and Hornik [40]), deep neural networks (DNNs) have been shown to outperform classical approaches (Bengio [41] and Kriegeskorte [42]). Although DNNs are mathematically and computationally more complex, their rise has been facilitated by advances in computational power and the availability of deep learning frameworks such as TensorFlow (Abadi et al. [43]) and PyTorch (Paszke et al. [44]). Consequently, DNNs have become suitable for a wide range of neuroscience applications (LeCun, Bengio, and Hinton [45]).

DNN-based models have been successfully applied to system identification tasks 2.1.1 in various stages of the visual pathway 1.2, including the retina (Maheswaranathan et al. [36]), primary visual cortex (V1) (Cadena et al. [46] and Kindel, Christensen, and Zylberberg [47]), and extrastriate regions (Zareh et al. [48]). Many of the current state-of-the-art methods use *convolutional neural networks*

(CNNs) (Krizhevsky, Sutskever, and Hinton [49]), which are particularly effective for visual tasks and have consistently outperformed classical models (Zhang et al. [50] and Cadena et al. [46]).

However, DNNs also have notable limitations. Chief among them is the large amount of data required to train these models effectively. In neuroscience, the quantity of experimental recordings from single neurons is often limited (Zhang et al. [50]). Some studies have addressed this by incorporating local receptive field constraints and jointly modeling multiple neurons (Antolík et al. [51]).

Recurrent Neural Networks

Another limitation of many state-of-the-art DNNs is their reliance on *feedforward* architectures, where information flows in a single direction. This contrasts with biological neural systems, which include recurrent (feedback) connections. Although feedforward models can perform well, enhancing biological plausibility and modeling temporal dependencies requires incorporating recurrent connections (Kafaligonul, Breitmeyer, and Öğmen [52], Shou [53], and Kar et al. [54]).

Recurrent neural networks (RNNs) (Medsker and Jain [55]) address this need by allowing connections that loop back to earlier layers, making them well suited for capturing temporal dynamics. RNNs have shown strong performance in modeling time-dependent processes in various brain systems (Mante et al. [56] and Song, Yang, and Wang [57]), and offer the flexibility to encode biologically inspired constraints at the single-neuron level (Mante et al. [56], Masse et al. [58], and Kim, Li, and Sejnowski [59]).

Thanks to advances in deep learning frameworks 2.2.2, RNNs have been used in a variety of tasks, from motor control (Sussillo et al. [60] and Saxena et al. [61]) to cognitive modeling (Masse et al. [58] and Goudar et al. [62]). However, RNN applications in visual system modeling remain relatively rare and CNNs continue to dominate the field. Only a few recent studies have introduced RNN architectures that rival CNN performance in visual tasks (Soo et al. [63]).

Interpretability of DNNs

As highlighted in Section 2.2.2, one of the major concerns with DNNs is their limited interpretability. Critics argue that DNNs merely replace simple black-box models with more complex ones, without providing new insights into neural computation. For a deeper discussion on this issue, we refer the reader to Kriegeskorte [42], which serves as a primary reference for this section.

In fact, accurate prediction of neuronal responses does not automatically equal a mechanistic understanding of neural processing. However, by capturing low-level dynamics, DNNs provide a platform for in silico experimentation that

can complement in vivo data, especially when experimental recordings are sparse or noisy.

Another concern lies in the complexity of DNNs. Their sheer number of parameters and lack of transparency can make it difficult to interpret their internal workings. Still, one could argue that, since neural processing is inherently complex, it may be unrealistic to expect fully intuitive, low-dimensional descriptions. Instead, our goal should be to build models that are as simple as possible but still rich enough to capture the relevant biological dynamics.

Furthermore, DNNs do not need to perfectly replicate biological circuits to be useful. A certain level of abstraction is expected and desirable in modeling. The key is to define which aspects of the biological system the model aims to capture and to evaluate the quality of the model accordingly. Increasing the biological plausibility of models, such as using RNNs for recurrent dynamics 2.2.2 can improve both interpretability and predictive power.

In summary, while DNNs come with challenges, especially regarding interpretability, they offer a powerful framework for computational neuroscience. With thoughtful application and biologically motivated design, DNNs can contribute meaningfully to our understanding of visual, motor, and cognitive functions.

2.2.3 Spiking Neural Networks

Thus far, our focus has primarily been on the predictive performance of models in visual system identification tasks 2.1.1, with limited emphasis on their biological plausibility, apart from a brief discussion of recurrent connections in Section 2.2.2. In this section, we turn our attention to a modeling approach that prioritizes biological realism: *spiking neural networks* (SNNs). These models aim to replicate the known biological mechanisms of neural processing and, along with CNNs 2.2.2, represent the state-of-the-art approach in visual system modeling.

We will introduce a foundational class of SNNs known as *integrate-and-fire models*, which are among the most widely used biologically inspired models of neural activity. Notably, the model proposed by Antolík et al. [10], which serves as the source of our artificial dataset and the reference behavior in this study, is based on this family.

Because these models seek to reflect biological processes, it is useful to revisit the relevant neurobiological concepts discussed in Chapter 1, particularly Section 1.1 on neuronal physiology. Furthermore, the modeling principles in computational neuroscience, reviewed in Section 2.1, form the theoretical foundation for this approach. For a general overview of neuroscience and computational modeling, the reader may consult classic texts such as Bear, Connors, and Paradiso [12], Dayan and Abbott [22], and Gerstner and Kistler [64].

Recent developments in SNN research have also explored training methodologies adapted for their non-differentiable nature. Unlike traditional deep learning models trained via backpropagation, SNNs often rely on surrogate gradient methods, biologically inspired learning rules such as Spike-Timing Dependent Plasticity (Bi and Poo [65] and Caporale and Dan [66]), or approximations to handle discrete spike events (Haslinger, Pipa, and Brown [67]). These approaches aim to reconcile the gap between biological plausibility and computational tractability, opening the door to more robust learning in spiking models. Moreover, hybrid models that integrate DNN structures with spiking neuron dynamics are beginning to emerge as promising solutions that balance interpretability, performance, and biological fidelity (Eshraghian et al. [68] and Lee, Delbruck, and Pfeiffer [69]).

Compared to conventional DNNs, SNNs provide a more direct link to the temporal and event-driven nature of biological neural processing. Although DNNs typically rely on continuous-valued activations and dense matrix operations, SNNs communicate through sparse, temporally precise spikes. This sparsity offers potential computational advantages, such as lower energy consumption and greater robustness to noise—features especially desirable in neuromorphic computing. However, DNNs currently maintain a performance advantage in many large-scale learning tasks, partly due to their compatibility with modern hardware and well-established training pipelines. Bridging this gap between biological plausibility and computational performance remains an active area of research (Eshraghian et al. [68]).

Integrate-and-Fire Models

A wide range of models have been developed to incorporate biological realism into neural simulations. For instance, the Hodgkin-Huxley model (Hodgkin and Huxley [70]) offers a highly detailed description of the ionic mechanisms that underlie action potentials 1.1.2. While this model is biologically accurate, its complexity often makes it analytically and computationally intractable.

To address this, simplified models such as the *integrate-and-fire* family have been introduced. These models abstract away spike shape and biophysical detail, representing action potentials as binary events: spikes occur when the neuron's membrane potential exceeds a threshold. The dynamics between spikes are governed by the evolution of the membrane potential, modeled using electrical circuit analogies.

In a basic integrate-and-fire model, the neuron membrane is conceptualized as a capacitor with capacity C in parallel with a resistor of resistance R_m. When a current $i(t)$ is injected (e.g., by synaptic input), the capacitor charges and the potential $u(t)$ across the membrane increases. Some of the current leaks through the resistor, mimicking the leaky nature of biological membranes.

This setup leads to a first-order differential equation:

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u(t) + R_m i(t) \tag{2.5}$$

where $\tau_m = R_m C$ is the membrane time constant and $u(t)$ is the membrane potential at time t.

Spikes are not explicitly modeled, but occur as threshold-crossing events when $u(t)$ reaches the threshold value $U_{\text{threshold}}$:

$$t^{(f)} : \quad u(t^{(f)}) = U_{\text{threshold}} \tag{2.6}$$

Upon firing, the membrane potential is instantaneously reset to the resting potential U_{rest}:

$$\lim_{\delta \to 0; \delta > 0} u\left(t^{(f)} + \delta\right) = U_{\text{rest}} \tag{2.7}$$

After this reset, the same membrane dynamics resume. The model output is a sequence of firing times $t_i^{(f)}$ for each neuron i, where $f = 1, 2, \ldots$ denotes the spike index.

This formulation is known as *leaky integrate-and-fire model*, one of the simplest and most commonly used SNN models. Although it omits details such as spike shape, refractory periods, and adaptation, it retains the essential temporal dynamics and threshold-based spiking behavior.

For readers interested in further exploring this class of models and their biological underpinnings, we recommend the foundational texts by Dayan and Abbott [22] and Gerstner and Kistler [64] that served as major source of information for this section.

2.3 Evaluation Methods

Up to this point, we have discussed key concepts and modeling strategies in computational neuroscience. However, we have not yet addressed how the performance of these models is evaluated. There is currently no universal consensus on a single best evaluation metric for visual system identification tasks, and as a result, a wide range of evaluation criteria have been developed. For an in-depth discussion of this issue, see Pospisil and Bair [71] and Carandini et al. [24].

In this section, we focus on two evaluation metrics that are central to our study: *Pearson's correlation coefficient* and *normalized cross-correlation.*

Neural responses are typically recorded as membrane potentials or time-varying firing rates (see Equation 2.1.3). Consequently, a performance metric should quantify the similarity between the predicted firing rate $\hat{r}(t)$ and the recorded rate $r(t)$ over time. Due to the trial-to-trial variability in neural responses, particularly in sensory systems, it is standard to compute average responses across trials. Since trial-to-trial variability can only be averaged asymptotically as the number of trials $N \to \infty$, practical evaluations must balance noise and data availability, which are often limited by the cost and complexity of experiments. It is also important to note that predictive accuracy alone does not guarantee meaningful model performance; multiple evaluation dimensions should be considered.

2.3.1 Pearson's Correlation Coefficient

One commonly used similarity measure in statistics is Pearson's Correlation Coefficient (PCC), which quantifies the linear relationship between two variables. The PCC between two random variables X and Y is defined as:

$$CC_{abs}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} \tag{2.8}$$

We denote the coefficient as CC_{abs} to distinguish it from the normalized version described later. For finite set of samples $\hat{X}$ and $\hat{Y}$ with $M \in \mathbb{N}$ samples, the sampled PCC is given by the following:

$$CC_{abs}(\hat{X}, \hat{Y}) = \frac{\sum_{i=1}^{M}(\hat{x}_i - \mu_{\hat{X}})(\hat{y}_i - \mu_{\hat{Y}})}{\sqrt{\sum_{i=1}^{M}(\hat{x}_i - \mu_{\hat{X}})^2}\sqrt{\sum_{i=1}^{M}(\hat{y}_i - \mu_{\hat{Y}})^2}} \tag{2.9}$$

where $\mu_{\hat{X}}$ and $\mu_{\hat{Y}}$ are the sample means.

For neuronal responses, we evaluate PCC between predicted and recorded average firing rates across trials for a given subset of neurons. Let $\hat{R}, R \in \mathbb{R}^{M \times T}$ be the matrices of the mean predicted and recorded responses, respectively, where M is the number of neurons and T is the number of time bins. We denote $\hat{R}(t)$ and $R(t)$ as the responses at time step t. Then, the time-resolved PCC is:

$$CC_{abs}(\hat{R}(t), R(t)) = \frac{\frac{1}{M}\sum_{i=1}^{M}(\hat{r}_i(\overline{t}) - \mu_{\overline{\hat{R}_t}})(r_i(\overline{t}) - \mu_{\overline{R_t}})}{\sqrt{\frac{1}{M}\sum_{i=1}^{M}(\hat{r}_i(\overline{t}) - \mu_{\overline{\hat{R}_t}})^2}\sqrt{\frac{1}{M}\sum_{i=1}^{M}(r_i(\overline{t}) - \mu_{\overline{R_t}})^2}} \tag{2.10}$$

A more compact representation, analogous to Wang et al. [72], is:

$$CC_{abs} = \frac{\mathrm{Cov}(\overline{\hat{R}}, \overline{R})}{\sqrt{\mathrm{Var}(\overline{\hat{R}})\mathrm{Var}(\overline{R})}} \tag{2.11}$$

Although PCC is generally computed between trials for a single time interval, we sometimes evaluate it over multiple time steps 2.1.3 by aggregating results across selected intervals.

PCC satisfies many desirable properties: it is bounded in $[-1, 1]$, easy to interpret, and relatively insensitive to sample size or bias (Wang et al. [72]). However, it does not account for trial-to-trial variability. Consequently, a low PCC may arise from noisy data rather than poor model performance. This limitation motivates the use of alternative measures such as normalized cross-correlation.

2.3.2 Normalized Cross Correlation

To address the limitations of PCC, Hsu, Borst, and Theunissen [73] proposed the *normalized cross-correlation* (NCC), which adjusts for the inherent variability of the neural data. This metric has since been used in several performance studies (Touryan, Felsen, and Dan [74], Gill et al. [75], and Wang et al. [72]) and is defined as:

$$CC_{norm} = \frac{CC_{abs}}{CC_{max}} \tag{2.12}$$

Here, CC_{max} represents the theoretical upper bound of PCC given perfect predictions, adjusted for neural variability. Although CC_{max} requires infinite trials for an exact computation, it can be approximated using finite data. The original formulation by Hsu, Borst, and Theunissen [73] was improved by Schoppe et al. [76], who introduced an efficient approximation applied in Wang et al. [72].

Using trial averages $\bar{\cdot}$ and neural response matrices $\hat{R}, R \in \mathbb{R}^{M \times T}$ as defined earlier, the generalized CC_{max} is:

$$CC_{max} = \sqrt{\frac{N\mathrm{Var}(\overline{R}) - \overline{\mathrm{Var}(R)}}{(N-1)\mathrm{Var}(\overline{R})}} \tag{2.13}$$

And in a more specific neuron-wise formulation:

$$CC_{max} = \sqrt{\frac{N\left(\frac{1}{M}\sum_{i=1}^{M}(\overline{r_i(t)} - \mu_{\overline{R_t}})^2\right) - \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{M}\sum_{j=1}^{M}(r_{ij} - \bar{r}_i)^2\right)}{(N-1)\left(\frac{1}{M}\sum_{i=1}^{M}(\overline{r_i(t)} - \mu_{\overline{R_t}})^2\right)}} \tag{2.14}$$

Although NCC is not perfect, it has been shown to outperform other evaluation metrics in certain contexts (Schoppe et al. [76] and Pospisil and Bair [71]), and we adopt it as a complementary measure in our own evaluation framework.

Chapter 3

Methods

In this chapter, we begin by describing the artificial stimulus dataset used to train our model, including a brief overview of the model from which the dataset originates. We then describe in detail the preprocessing steps applied to the dataset to make it compatible with our model. The final and major section of this chapter focuses on our biologically plausible recurrent neural network (RNN) model of the primary visual cortex (V1) and its additional extensions.

3.1 Spiking Model of Cat Primary Visual Cortex

A major challenge in computational neuroscience, particularly in system identification tasks (Section 2.1), is the limited availability and quality of biological data. This issue becomes even more problematic when developing models using deep neural networks (DNNs), as discussed in Section 2.2.2. The noise and expense of biological data acquisition motivate the use of artificial datasets. In our work, we utilize data generated by a state-of-the-art spiking model of the cat V1 developed by Antolík et al. [10].

This thesis focuses exclusively on developing novel RNN methods using the synthetic data produced by highly biologically detailed models of primary visual cortex described below. As we will see, this a very challenging problem even when using the synthetic data. However the long-term objective is to translate the data to be applicable to parallel population recordings in-vivo, which is however beyond the scope of this thesis.

3.1.1 Spiking Model Description

The artificial dataset is derived from a biologically detailed spiking model of cat V1 introduced by Antolík et al. [10]. Designed to simulate visual experiments,

this model aims to address the challenges inherent in biological data acquisition.

The model is implemented as a spiking neural network (SNN) using the exponential integrate-and-fire neuron model (Fourcaud-Trocmé et al. [77]), an extension of the leaky integrate-and-fire model (Section 2.2.3). It captures the spatial layout of cortical layers IV and II/III in a 5.0 x 5.0 mm patch corresponding to the *area centralis* of the retina, responsible for high-acuity vision (Section 1.3.1 and Section 1.5).

The model incorporates various biological properties, such as differentiation between excitatory and inhibitory neurons, synaptic depression, and local connectivity among neurons with similar orientation preferences (Section 1.1.3 and Section 1.5.1). It has been validated against a wide range of experimental data, including responses to white-noise and natural stimuli (Section 2.1.2). Compared to other models, it is distinguished by its comprehensive inclusion of biological features (Antolík et al. [10]).

A particularly notable strength of the model is its ability to reproduce spontaneous neuronal activity without artificial noise sources. This enables it to replicate trial-to-trial variability in stimulus responses (Section 2.3), as demonstrated in Baudot et al. [78]. These capabilities were validated using blank gray stimuli (Papaioannou and White [79]), making the model a valuable source of biologically plausible data.

Despite its strengths, the model remains a simplification of the biological system. For instance, feedback from cortical layers V and VI and the diversity of neuron types within layers are not explicitly modeled (Section 1.5). Furthermore, the recurrent pathway from layer II/III to IV serves as a representative for these excluded modulatory circuits.

Model Architecture

The model architecture is profoundly inspired by the biological organization of the primary visual cortex (V1), as described in detail in Section 1.5. It is organized into three main neuronal populations, corresponding to the LGN neurons, layer IV neurons, and layer II/III neurons of V1. Within Layers IV and II/III, neurons are further subdivided into two functional groups: inhibitory neurons, which form connections exclusively within their own layer, and excitatory neurons, which project both within their layer and across layers.

Specifically, excitatory neurons in Layer IV project to both excitatory and inhibitory neurons in Layer II/III, while excitatory neurons in Layer II/III project back to both excitatory and inhibitory neurons in Layer IV. This feedback connection serves as a proxy for biological feedback originating from deeper cortical layers (V and VI), as previously discussed. Additionally, Layer IV receives input from LGN neurons, with ON and OFF pathways differentiated to reflect the func-

tional organization of early visual processing. The overall architecture of the SNN model is illustrated in Figure 3.1, taken from Antolík et al. [10].

The model represents a substantial downsampling relative to the biological system. It comprises 108,150 neurons and approximately 155 million synapses, capturing around 10% of the neuronal density found in the cat V1 (Beaulieu and Colonnier [80]). Neurons are distributed across layers according to biological proportions, maintaining a 4:1 ratio of excitatory to inhibitory neurons (Beaulieu et al. [81] and Markram et al. [82]).

Furthermore, a simplified lateral geniculate nucleus (LGN) model is incorporated (Section 1.4), featuring distinct ON and OFF pathways, as shown in Figure 1.4 and Figure 1.6. These 7,200 LGN neurons are spatially aligned with the receptive fields of their corresponding V1 neurons. For additional details, refer to Antolík et al. [10].

3.2 Artificial Dataset

Our dataset consists of spiking activity generated by the model described above in the Section 3.1. It includes responses from 14,400 LGN neurons and 108,150 V1 neurons, recorded at 1 ms resolution. The data is organized into sequences corresponding to multiple experiments, each comprising alternating blank and natural image stimuli. Due to the model's comprehensiveness, unnecessary information was removed during preprocessing, as described below.

3.2.1 Stimulus Sequence Structure

The dataset is divided into *stimulus sequences*, each comprising alternating intervals of a blank gray screen and a natural stimulus image neuronal responses. Each pair forms a *stimulus pair*, where the blank interval precedes the stimulus to ensure that only spontaneous activity is present before the image is shown.

Each stimulus sequence begins with a blank interval of duration $t_{\text{blank}} \in \mathbb{N}$, followed by a stimulus interval of duration $t_{\text{stim}} \in \mathbb{N}$. These pairs are repeated periodically. Blank intervals serve to reset neural activity, preventing carryover effects from previous stimuli. Notably, the final blank interval following the last stimulus is omitted, introducing a minor inconsistency.

3.2.2 Experiment Definition

To facilitate processing and reduce memory demands, stimulus sequences are divided into smaller units referred to as *experiments*. Each experiment consists of a natural stimulus sequence immediately followed by a blank stimulus interval.

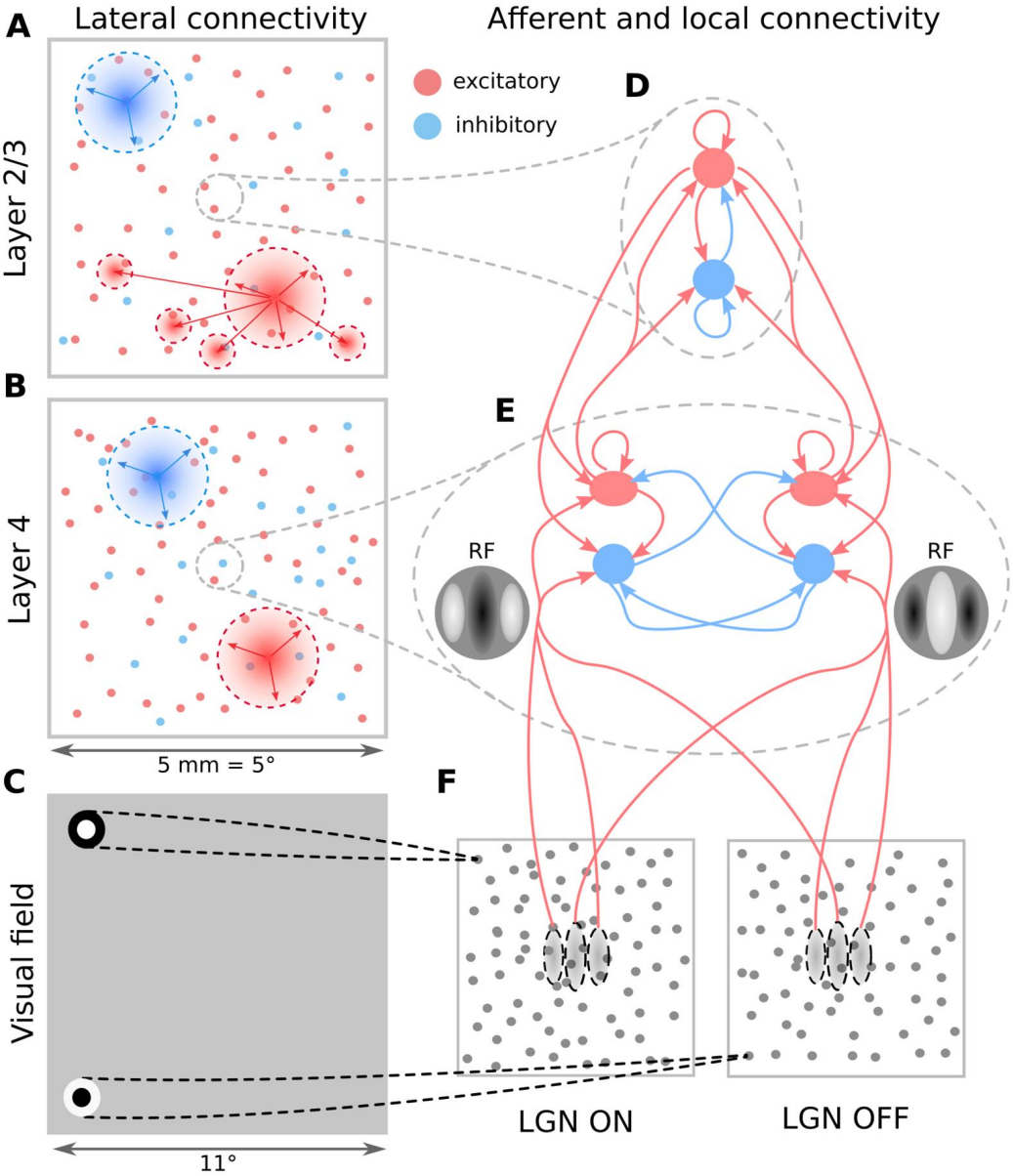

Figure 3.1 Architecture of the SNN Model of Cat's V1: This figure illustrates the architecture of the spiking neural network (SNN) model of the cat's primary visual cortex (V1) developed by Antolík et al. [10]. (A-B) Lateral connectivity within Layers II/III and IV. (C) Extent of the modeled visual field and examples of receptive fields (RFs) for one ON and one OFF LGN neuron. (D-E) Local connectivity schemes within layer II/III and layer IV. (F) Input connections from the LGN to Layer IV. This figure is taken from *"A comprehensive data-driven model of cat primary visual cortex"* by Antolík et al. [10].

This design ensures that the neural activity at the boundaries of each stimulus reflects a spontaneous, baseline state.

Definition 1 (Experiment)**.** *Let S be a stimulus sequence consisting of $N \in \mathbb{N}$ pairs of blank and stimulus intervals (b_i, s_i). An experiment for each pair index $j \in \{1, 2, \ldots, N-1\}$ is defined as:*

$$\left(s_j; b_{j+1}\right)$$

The final experiment omits the trailing blank interval:

$$\left(s_N\right)$$

To maintain consistency, the initial blank interval preceding the first stimulus is omitted. While this results in minimal data loss, it simplifies the formatting and processing of experimental sequences. For cases where a post-stimulus blank interval is missing, zero-padding is applied to ensure uniformity.

This splitting strategy is motivated by the original design of the stimulus sequence. The inclusion of blank intervals aims to capture spontaneous neuronal activity and reset neural responses to a baseline state uninfluenced by preceding stimuli. Based on this rationale, we empirically assert that starting each experiment with a natural stimulus should consistently provide neuronal responses representative of spontaneous activity. Consequently, partitioning the sequence into smaller experiments should not result in significant information loss compared to passing the entire stimulus sequence at once.

3.2.3 Dataset Preprocessing

The original dataset includes detailed simulation data, much of which is irrelevant for training. We extract only the spike trains (Section 2.1.3) and segment them into experiments, reducing memory and computation demands significantly.

To streamline this process, we have developed a dedicated data processing pipeline that converts raw simulation output into the final experiment-based format used by our model. This pipeline is modular and reusable, allowing efficient processing of future datasets generated by the same spiking model. As such, it facilitates further experiments and model iterations without redundant implementation effort.

3.2.4 Dataset Description

After the raw dataset generated from the model by Antolík et al. [10] undergoes the preprocessing steps described in Section 3.2.3, we obtain the final dataset used

in our model. The basic unit of this dataset is the experiment (Section 3.2.2). Each experiment is represented by six matrices corresponding to different neuronal populations. Two matrices represent the spike trains of ON and OFF LGN neurons (denoted as s_{ON} and s_{OFF}), while the remaining four correspond to layer IV and layer II/III neurons, further divided into excitatory and inhibitory types: r_{E_4}, r_{I_4}, $r_{E_{2/3}}$, and $r_{I_{2/3}}$. The architecture and number of neurons in each population is defined in Section 3.1.1, matching the configuration in Antolík et al. [10].

Each matrix has the format $\mathbb{R}^{N \times M \times T}$, where N is the number of experiments, M is the number of neurons in the given population, and T is the number of time steps in each experiment (see Section 3.2.2 for experiment definition).

In our artificial dataset, the blank stage lasts for 150 ms, and the natural stimulus stage lasts for 560 ms, both recorded in 1 ms bins. Thus, the spike train for a single neuron in an experiment spans 710 ms, composed of 75 ms of a blank interval before the stimulus, 560 ms of stimulus presentation, and 75 ms of the subsequent blank interval. The spike train is encoded in binary format: a 1 indicates a spike at a given time step, and a 0 indicates no spike. For experiments at the end of a stimulus sequence, the final blank interval is missing. To ensure uniform experiment duration, we pad these with zeros (no spikes).

Although this padding does not accurately reflect biological spontaneous activity (which is non-zero, see Section 3.1.1), it affects only a tiny minority of cases. We chose this pragmatic solution for its simplicity and minimal impact on model performance.

In classical machine learning, datasets are typically split into training, validation, and test subsets, often with randomized sampling. However, our use case is complicated by trial-to-trial variability in neuronal responses, necessitating multiple repeated trials in the validation and test datasets (see Section 2.3). Because this variability arises mainly from spontaneous neural activity (Antolík et al. [10]), we chose not to include multiple trials in the training set. Instead, we prioritized generating a wider variety of single-trial responses to diverse natural stimuli. This design choice balances several practical factors: dataset size, memory efficiency, training speed, and simulation time.

Since the training and test datasets were generated separately, we do not perform randomized splitting across the full dataset. Instead, we rely on a single, fixed variant for both training and evaluation/test, as alternative splits would not be suitable for this setup.

Train Dataset

The training dataset consists of 500 stimulus sequences, each containing 100 pairs of blank and natural stimuli, all presented in a single trial. After preprocessing and experiment segmentation (Section 3.2.2), this yields a total of 50,000 unique

experiments for training.

Test and Validation Dataset

The validation and test datasets comprise 18 stimulus sequences, each with 50 stimulus pairs repeated over 20 trials. After preprocessing, this results in 900 experiments, each with 20 trial repetitions. We randomly divide this set into validation and test subsets using a 1:8 ratio.

Neuron Subset Selection

To build a biologically plausible RNN model that mirrors the structure in Antolík et al. [10], one would ideally create a one-to-one mapping between model neurons and SNN neurons. However, given the lack of structural constraints (e.g., known spatial synaptic constraints) and the impracticality of modeling over 100,000 neurons, we adopt an all-to-all connectivity scheme with neuron subsampling.

We randomly select 10% of neurons from each population to reduce model complexity and memory consumption, making the model more tractable during training and fine-tuning. To mitigate sampling bias, we run experiments using 20 different random neuron subsets. Empirical testing has shown that using this 10% subset produces performance metrics similar to those obtained with larger neuron subsets. Further evaluation of this design choice is presented in the next chapter.

3.3 Model Description

This section describes the core model used in our study, followed by the biologically inspired enhancements we applied to improve its performance.

Our goal is to develop a model for visual system identification task, as introduced in Section 2.1. Specifically, we develop a recurrent neural network (RNN) that takes in time-series data from LGN neurons and predicts the activity of selected V1 neurons. Both input and target output data come from the artificial dataset described in Section 3.2.

At every time step $t \in \mathbb{N}$ in the experimental sequence (Section 3.2.2), the model receives a vector of inputs representing responses from LGN ON and OFF cells. We denote this vector as $s(t)$:

$$s(t) = (s_{ON}(t); s_{OFF}(t))$$

Our objective is to learn a biologically plausible function f implemented via an RNN, which maps the LGN input to predicted V1 responses:

$$\hat{r}(t) = f(s(t))$$

The output vector $\hat{r}(t)$ includes predictions for four types of V1 neuron populations:

$$\hat{r}(t) = \left(\hat{r}_{E_4}(t); \hat{r}_{I_4}(t); \hat{r}_{E_{2/3}}(t); \hat{r}_{I_{2/3}}(t)\right)$$

In this text, we use a hat symbol ($\hat{\cdot}$) to indicate predicted values, while unmarked variables refer to the ground truth values from our dataset.

The main learning goal is to minimize the difference between predicted responses $\hat{r}(t)$ and actual responses $r(t)$, while accurately modeling the temporal dynamics of each neuron. As noted in Section 2.3, there is no universally accepted evaluation metric for this problem. For our analysis on the test dataset (Section 3.2.4), we use Pearson's correlation coefficient (Section 2.3.1) and normalized cross-correlation (Section 2.3.2) to evaluate model performance.

For training, we use the mean squared error (MSE) loss function (Alpaydin [30]). The model is trained by minimizing the following objective function during backpropagation:

$$L(\hat{R}, R) = \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\hat{R}_i(t) - R_i(t)\right)^2$$

Here, $\hat{R}, R \in \mathbb{R}^{N \times M_{out} \times T}$ represent the predicted and actual response matrices, where N is the number of training experiments, M_{out} is the number of output neurons, and T is the total number of time steps.

Choosing an appropriate loss function for visual system identification is still an open question in the field. We chose MSE because it is widely used, easy to interpret, and simple to implement. It has also been effective in many machine learning and signal processing applications (Wang and Bovik [83] and Söderström [84]), including prior neuroscience modeling studies (Antolík et al. [51]).

That said, other loss functions such as the Poisson loss are often more suitable for modeling neural spike data, since these outputs represent discrete counts. Several recent studies (Terven et al. [85], Wang et al. [72], and Sinz et al. [86]) have used Poisson loss for this reason. Although Poisson loss may provide more biologically accurate modeling, we chose MSE due to its strong performance in our early trials of research and the practical advantages it offers. We recognize that this choice may have some limitations and suggest testing alternative loss functions in future research.

To optimize the model, we use the Adam optimizer (Kingma and Ba [87]), a widely used method in deep learning known for its fast convergence and reliability.

3.3.1 Base Model Architecture

This section introduces the base model architecture, which serves as the foundation for later enhancements aimed at improving both model performance and biological realism.

We refer to this initial version as the *simple model*, and it acts as the starting point for further development throughout this study.

The simple model is a recurrent neural network (RNN) designed to approximate the structure and function of layers IV and II/III of the primary visual cortex (V1). Its design is to directly mimic the architecture of the spiking neural network (SNN) described in Antolík et al. [10], which also provides the artificial dataset used for training and evaluation. Because fully replicating all biological details is computationally intensive and beyond the scope of this thesis, we introduce several simplifications that maintain key structural aspects while keeping the model tractable. Our long-term objective is to integrate this RNN with the original SNN to build a more comprehensive simulation framework.

The model takes as input a sequence of spiking responses from LGN neurons, represented by vectors s_{ON} and s_{OFF}, corresponding to ON and OFF LGN cell populations. These inputs are fed into two RNN layers that model the excitatory (E_4) and inhibitory (I_4) subpopulations of layer IV in V1. Each artificial neuron is designed to correspond to a neuron in the original SNN (which implies the same for the biological neuron), though we use only a subset of these due to memory constraints (Section 3.2.4).

Within layer IV, E_4 and I_4 are fully interconnected and also include full self-connections. The output from E_4 is passed to two additional RNN layers that represent the excitatory ($E_{2/3}$) and inhibitory ($I_{2/3}$) subpopulations of layer II/III. These layers follow the same full connectivity and self-connectivity pattern as layer IV. Additionally, we include a recurrent feedback connection from $E_{2/3}$ to both E_4 and I_4. This simplifies the feedback typically mediated by deeper cortical layers (V and VI), as described in Antolík et al. [10] and Section 1.5.

We define the operation of each RNN layer as follows:

Definition 2 (Base Neuronal Layer)**.** *Let L be a layer with $m_h \in \mathbb{N}$ neurons. Let $x \in \mathbb{R}^{m_{in}}$ be the input vector, where $m_{in} \in \mathbb{N}$ is the number of input neurons. Let $W_{ih} \in \mathbb{R}^{m_h \times m_{in}}$ and $b_{ih} \in \mathbb{R}^{m_h}$ be the input weight matrix and bias vector. Let $h \in \mathbb{R}^{m_h}$ be the previous hidden state of the layer, with recurrent weights $W_{hh} \in \mathbb{R}^{m_h \times m_h}$ and bias $b_{hh} \in \mathbb{R}^{m_h}$. Let $f : \mathbb{R}^{m_h} \to \mathbb{R}^{m_h}$ be an activation function. Then the updated hidden state h' is calculated as:*

$$h' = f(W_{ih}x + b_{ih} + W_{hh}h + b_{hh})$$

Note: *The input vector x aggregates neuronal responses from all input layers*

connected to the target layer. These inputs may include both the responses from LGN neurons (designated as input layer neurons) and from other layers that project to the target layer.

The activation function f can be any non-linear function that operates element-wise. In this thesis, we often use vector notation for clarity. Unless otherwise stated, f is applied independently to each element of the vector.

Additionally, for each pair of input and output layer has its own synaptic depression function that modifies the input before it is passed for processing for the layer. This function is identity in the base model but will be extended and closely described in the following sections on additional modules. These modules are applied before each input is passed (not for the outputs of the model).

The high-level architecture of the model is illustrated in Figure 3.2.

This model architecture is loosely based on the biological structure of V1 (Section 1.5) and incorporates similar simplifications found in the original SNN (Antolík et al. [10]). Key simplifications made relative to the SNN model include:

- Modeling only a randomly chosen subset of neurons, not the full population.
- Using all-to-all connectivity rather than spatially constrained, probabilistic connections.

In contrast, the original SNN uses a connection probability that decreases with spatial distance and biases connections based on features like orientation selectivity. While the all-to-all connnections simplify the architecture of the RNN mode, they greatly increase number of parameters and reduce biological realism. Future work could reintroduce spatial constraints to make the model more scalable and biologically accurate.

At each time step t, the model's output consists of the concatenated outputs from all four RNN layers:

$$\hat{R}(t) = (E_4(t); I_4(t); E_{2/3}(t); I_{2/3}(t))$$

Time Bins Merging

Training neural networks with millisecond-resolution data is both time- and memory-intensive, especially when using single-trial recordings, which are inherently noisy. After inspecting the dataset and considering the trade-off between temporal detail and computational efficiency, we decided to aggregate the data into 20 ms bins instead of 1 ms. This change preserves meaningful temporal patterns in neural activity while significantly reducing data volume and smoothing out noise. While this binning reduces high-frequency detail, our data showed low spike rates overall, making this trade-off acceptable. The impact of this decision is further discussed in the results section.

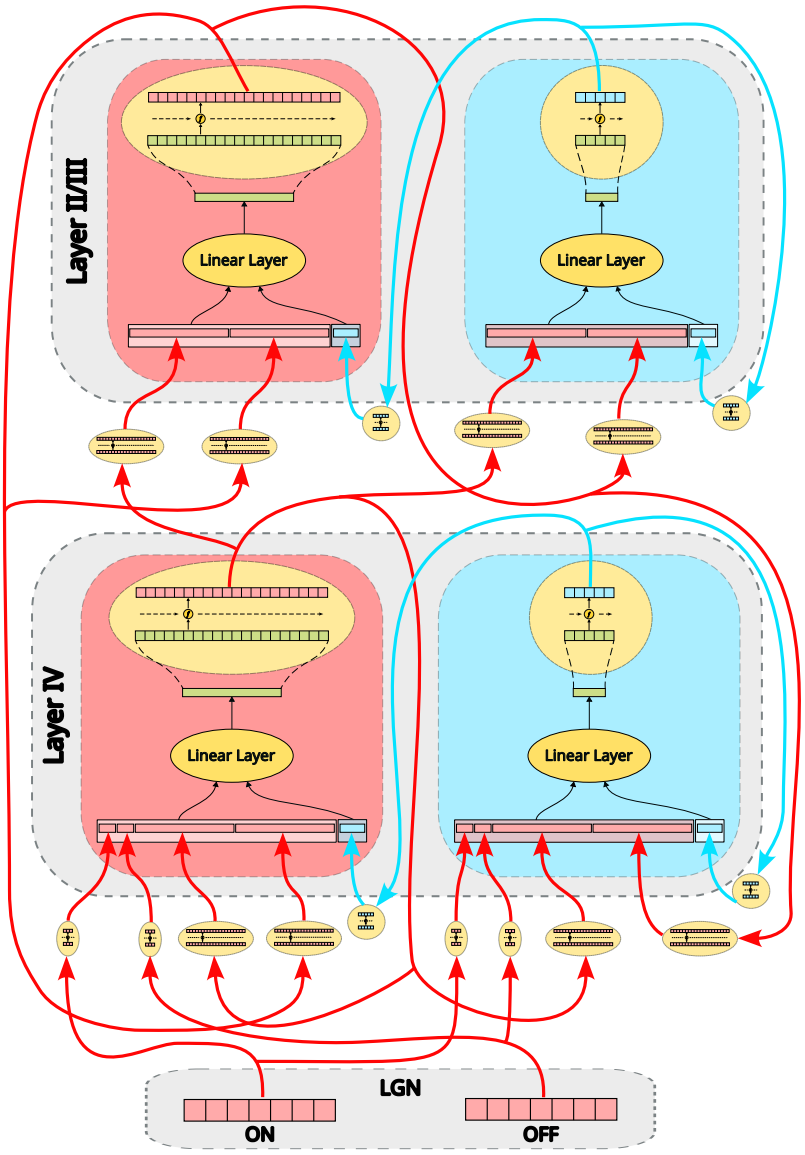

Figure 3.2 Overview of the model architecture. This figure presents the overall structure of the RNN model for V1. Grey modules represent different layers of the model. Red and blue highlights denote components belonging to excitatory and inhibitory layers, respectively. Arrows indicate the flow of inputs, bars represent specific input or output values, and the large colored regions within layers illustrate modules corresponding to either excitatory or inhibitory neuronal populations. Each excitatory or inhibitory module first concatenates inputs from all incoming layers, passes them through a linear layer, and then applies an element-wise activation function (referred to as the "neuron module" in subsequent sections). Additionally, each input pathway is associated with a synaptic depression modules, which modifies the input before it reaches the corresponding neuronal population module.

LeakyTanh Activation Function

As mentioned in Definition 3.3.1, the model architecture allows flexibility in choosing the activation function. For our baseline RNN, we use a custom activation function called *LeakyTanh*, which was developed by Richard Kraus as part of his bachelor's thesis, which extends this work.

LeakyTanh was designed to produce biologically plausible outputs while preserving stable gradients during training. The goal was to create an activation function suitable for predicting firing rates, which should be non-negative. Given our 20 ms binning and the theoretical upper bound of 20 spikes per bin (based on a maximum rate of one spike per millisecond (Dayan and Abbott [22])), it makes sense for the activation function to output values in a narrow, biologically meaningful range. However, our dataset rarely shows more than four spikes per bin, and most responses are binary, so even lower outputs are typically sufficient.

We tested several activation functions, including ReLU, tanh, and custom variants. LeakyTanh provided the most stable training behavior. We believe this is because it is fully differentiable across $\mathbb{R}$ and unbounded on the upper end, while still being partially constrained from below. Functions that are bounded on both ends performed poorly (Dubey, Singh, and Chaudhuri [88] and Nwankpa et al. [89]), and fully unbounded functions caused issues with exploding gradients, even when gradient clipping was applied. Although we still use gradient clipping with LeakyTanh, its design generally avoids the need for it.

The LeakyTanh function is defined as follows:

Definition 3 (LeakyTanh)**.** *Let* $[0, m]$ *be the target output range. Define* $c = \frac{m}{2} \in \mathbb{R}$ *as the central offset,* $s \in \mathbb{R}$ *as the steepness parameter for the* tanh *core, and* $l \in \mathbb{R}$ *as the leakage factor. We also use the softplus function* $softplus(x, \beta)$ *with steepness parameter* $\beta \in \mathbb{R}$ *as a leakage factor. The LeakyTanh function is defined as:*

$$LeakyTanh(x; c, s, l, \beta) = c + c \cdot \tanh(s \cdot x) + l \cdot softplus(x, \beta)$$

In our experiments, we used $c = 2.5$ (centering outputs around the 0–5 range), $l = 0.001$, $s = 0.2$, and $\beta = 20$. These parameters were fine-tuned by Richard Kraus. Our contribution focused on analyzing the dataset to define the desired output characteristics and comparing the performance of various activation functions in our model.

Excitatory/Inhibitory Neurons Differentiation

To increase biological realism, we differentiate excitatory and inhibitory neurons (Section 1.1.3) by applying sign constraints to their weights. Neurons in excitatory layers (E_4 and $E_{2/3}$) are constrained to have non-negative weights, promoting

excitatory (amplifying) effects. Inhibitory layers (I_4 and $I_{2/3}$) are constrained to have non-positive weights, modeling their suppressive role in the network.

These constraints are enforced after every optimization step. If any updated weight violates the constraint, it is clipped to the appropriate boundary.

Formally, the constraints are applied as follows:

Definition 4 (Weight Constraints)**.** *Let $w_E, w_I \in \mathbb{R}$ be weights from excitatory and inhibitory neurons, respectively. After each update step, we apply the following rules:*

Excitatory neurons:

$$w_{E_{new}} = \begin{cases} w_E & \text{if } w_E \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Inhibitory neurons:

$$w_{I_{new}} = \begin{cases} w_I & \text{if } w_I \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Training of the Model

The training procedure was conducted as follows. At each time step, the model receives the relevant input values and begins processing in layer IV. Because this layer also depends on recurrent inputs from other cortical layers, we provide these recurrent inputs using values from previous time steps of our artificial dataset. This approach is applied consistently across all recurrent connections in the model.

For each time step, we perform a forward pass through the network, compute the loss between the predicted and target neuronal responses at every layer, and then carry out the backpropagation and optimizer steps. After the model weights are updated, we apply weight constraints to maintain excitatory/inhibitory differentiation, as described in Section 3.3.1.

Importantly, we do not use the model's outputs as recurrent inputs for the following time step. Instead, we reset these inputs using the target values from the dataset. This ensures the model receives the correct inputs at each step and accelerates training. Since the simple model has no hidden internal states and we know how each neuron is expected to respond to all stimuli, we can take advantage of this unusual property of RNN training to simplify and stabilize learning.

We also experimented with a variant involving multiple forward steps over a sequence of inputs referred to here as a "hidden time step" approach where

the final output is used to make a prediction for a specific time point. This allows for backpropagation through time (BPTT) (Werbos [90]), enabling the model to develop a form of temporal memory. However, this approach did not yield meaningful improvements. A likely reason is that our model already has access to many ground-truth target time steps, making the introduction of additional predictive time steps unnecessary and potentially noisy. Moreover, while increasing the sequence length could theoretically help, it would require returning to finer temporal resolution, which as previously discussed introduces significant noise and computational cost (Section 3.3.1). For these reasons, we decided not to include this variant in the final version of the model, though it may prove valuable in future work.

Evaluation of the Model

The model evaluation procedure is straightforward. We initialize the recurrent inputs using the target values from the first time step in the artificial dataset. We then perform forward passes for each subsequent time step, using only the LGN input without resetting the recurrent inputs. This setup allows us to assess whether the model can successfully predict the entire sequence of neuronal responses in an experiment despite never being exposed to continuous sequences in this way during training.

3.3.2 Additional Modules

Beyond the simple model described in Section 3.3.1, we introduced several additional modules aimed at enhancing model performance and incorporating further biological constraints. These modules were added sequentially, each one motivated by the limitations of the previous version. This section presents the modules in the order they were added, allowing readers to follow the development process and understand the reasoning behind each addition. While each module is theoretically optional, they were designed to build upon one another, and omitting earlier components typically undermines their intended effect. All these variants are depicted in the Figure 3.2, Figure 3.3and Figure 3.4.

Feed-forward Neuron Module

The first and arguably most impactful extension was replacing the LeakyTanh activation functions in the simple model with small, trainable feed-forward neural network (DNN) modules. These modules aim to more accurately reflect the complex, nonlinear behavior of biological neurons.

We implemented four distinct DNN modules one for each output neuronal population (E_4, I_4, $E_{2/3}$, $I_{2/3}$) to capture functional differences not only between excitatory and inhibitory neuron types, but also across different cortical layers (layer IV vs. layer II/III) of the V1. To preserve the output properties described in Section 3.3.1, each DNN module ends with a LeakyTanh activation.

Definition 5 (DNN Joint Neuron Module)**.** *Let $n \in \mathbb{N}$ be the number of layers in the module and $s \in \mathbb{N}$ the size of each hidden layer. Each layer consists of a sequence of Layer Normalization (Ba, Kiros, and Hinton [91]), a fully connected layer, and a ReLU activation. The first layer differs slightly in that it consists only of a fully connected layer that maps the input (a scalar output from the base model defined in Definition 3.3.1) to the hidden size s. The complete module $f : \mathbb{R} \to \mathbb{R}$ is defined by the following sequence:*

1. *Fully connected layer: input size 1, output size s.*
2. *$(n-1)$ repetitions of:*
   (a) *Layer Normalization*
   (b) *Fully connected layer: input and output size s.*
   (c) *ReLU activation*
3. *Fully connected layer: input size s, output size 1.*
4. *Final LeakyTanh activation.*

Optionally, a residual connection may bypass the entire module and be summed with its output before the final activation.

We refer to the version of the model incorporating this module as the *DNN joint model*. The term "joint" reflects the naming convention used in our source code and indicates that the model jointly processes both excitatory and inhibitory outputs from the base RNN (defined in Section 3.3.1) through a shared DNN architecture. This contrasts with the "separate" model variant (described in the following section), which handles excitatory and inhibitory outputs using distinct pathways. The structure of the joint variant is shown in Figure 3.3, and the architecture of the DNN neuron module is illustrated in Figure 3.4.

The main motivation behind this extension is to capture the complex and diverse behavior of different neuronal subtypes and cortical layers more effectively. Additionally, it enables the network to learn more expressive, trainable activation functions in place of static ones typically used in deep learning, potentially improving model capacity and flexibility.

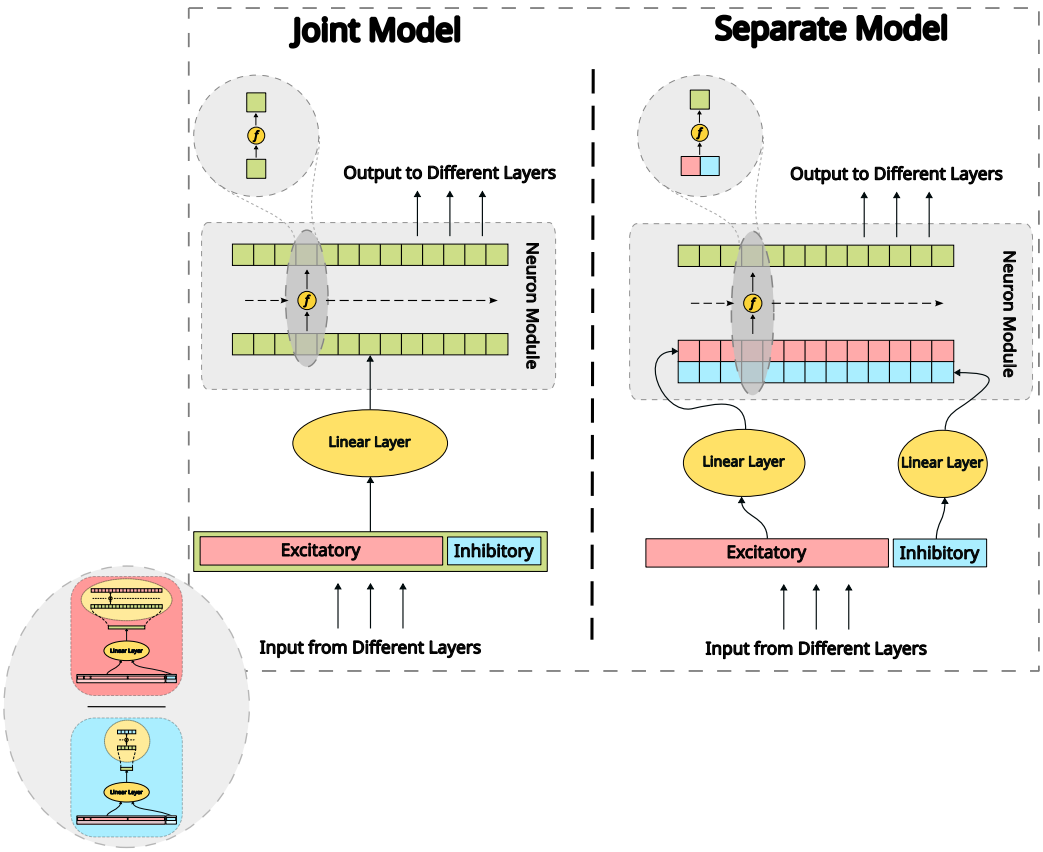

Figure 3.3 Schema of the joint and separate model types. This figure illustrates the information processing steps for both the joint and separate model variants. Green bars represent results combined from both excitatory and inhibitory layer data, while red and blue bars represent outputs from exclusively excitatory and inhibitory data, respectively.

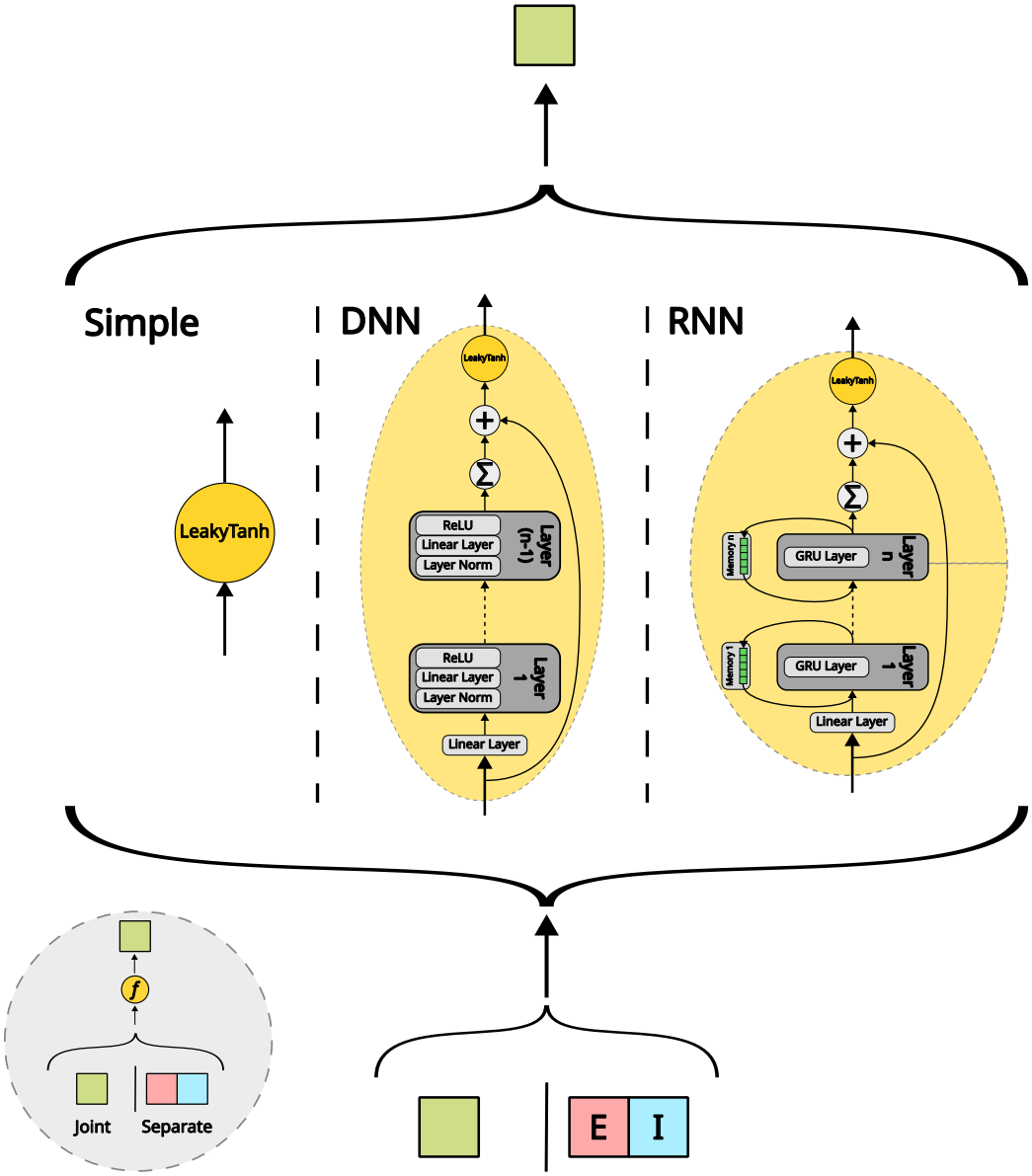

Figure 3.4 Neuron module processing types. This figure highlights the differences between the various neuron module types. The ”simple” module passes information through a leakytanh activation function, the ”DNN” module processes it through a feed-forward neural network, and the ”RNN” module passes it through a layered recurrent network (using a GRU variant in this case). The input structure is essentially the same for both the joint and separate model variants, with differences primarily in input size.

Splitting Excitatory/Inhibitory Output in Base RNN

In this version, the linear base RNN cell computation is split into two parts—one processing excitatory inputs and the other processing inhibitory inputs. The output of the base RNN neuron is thus a two-element vector (excitatory and inhibitory contributions), which is then passed to a shared DNN module, similar to that used in the DNN joint model (Definition 3.3.2). The main difference is that this module now takes an input of size 2. In the case of a residual connection, the excitatory and inhibitory inputs are summed before being added to the module output prior to the final activation function.

The purpose of this extension is to allow the model to distinguish and independently process excitatory and inhibitory contributions at the neuronal level.

Definition 6 (Separate Base RNN)**.** *Let L be a neuronal layer with $m_h \in \mathbb{N}$ neurons. Let $x_E \in \mathbb{R}^{m_{in_E}}$ and $x_I \in \mathbb{R}^{m_{in_I}}$ be input vectors from excitatory and inhibitory populations, respectively. Let W_{ih_E}, W_{ih_I} be their corresponding input weight matrices, and b_{ih_E}, b_{ih_I} be the associated bias vectors. Let $h \in \mathbb{R}^{m_h}$ be the recurrent input with weight matrix W_{hh} and bias b_{hh}. Let $f : \mathbb{R}^{2\times m_h} \to \mathbb{R}^{m_h}$ be the activation function. Then the output h' at the next time step is defined as:*

$$h' = \begin{cases} f\big(W_{ih_E}x_E + b_{ih_E} + W_{hh}h + b_{hh};\ W_{ih_I}x_I + b_{ih_I}\big) & \text{if } L \text{ is excitatory} \\ f\big(W_{ih_E}x_E + b_{ih_E};\ W_{ih_I}x_I + b_{ih_I} + W_{hh}h + b_{hh}\big) & \text{if } L \text{ is inhibitory} \end{cases}$$

The structure of the separate variant is shown in Figure 3.3.

As with the DNN joint model, we use the same shared DNN module for the activation function, except now it takes two inputs and handles them accordingly. Since this is a minor structural modification, we omit the formal redefinition.

Recurrent Neuron Module

We also implemented a recurrent version of the DNN neuron module, called either *RNN joint* or *RNN separate*, depending on whether it uses the joint (Definition 3.3.1) or separate (Definition 3.3.2) base model.

Instead of feed-forward layers, this version replaces the neuron modules with small recurrent networks (either LSTM (Hochreiter and Schmidhuber [92]) or GRU (Cho et al. [93])) to provide memory across time steps.

Definition 7 (RNN Neuron Module)**.** *Let $n \in \mathbb{N}$ be the number of RNN layers and $s \in \mathbb{N}$ the size of each hidden layer. Let $s_i \in \{1, 2\}$ be the input size (1 for joint, 2 for separate). Let $m_h \in \mathbb{N}$ be the size of the hidden state. The sequence of operations is as follows:*

1. *Fully connected layer: input size s_i, output size m_h.*

2. *Apply n-layer RNN (LSTM or GRU) to produce $h_{out} \in \mathbb{R}^{m_h}$ (final hidden state).*

3. *Fully connected layer: input size m_h, output size 1.*

4. *Final LeakyTanh activation.*

A residual connection may be added before the final activation.

The RNN neuron module is shown in detail in Figure 3.4.

This extension is motivated by the biological reality that neurons adapt based on short-term input history. Thus, temporal memory might help improve prediction of network dynamics.

Training this model requires adjustments to the procedure described in Section 3.3.1. Unlike the base model where hidden states are known from the dataset these neuron modules have internal hidden states that must be learned. Therefore, we introduce backpropagation through time (BPTT) (Werbos [90]).

To manage this, we use truncated BPTT (TBPTT) (Williams and Peng [94]). Specifically, we:

- Accumulate forward steps for a fixed window $T_{\text{back}} \in \mathbb{N}$.
- Perform the backward and optimizer steps after each window.
- Do not reset the internal RNN neuron states, but continue resetting the base RNN states with known values.

While TBPTT allows training, it increases memory and computation costs significantly. Thus, this module would benefit from future integration of spatial constraints to reduce parameter count—similar to those used in the original SNN (Antolík et al. [10]).

Synaptic Depression

Our final extension introduces synaptic depression (Abbott et al. [14]), mimicking short-term plasticity effects like reduced neurotransmitter release under sustained firing. This phenomenon is already represented in the original template model (Antolík et al. [10]).

We model synaptic depression using small shared RNN modules placed between presynaptic and postsynaptic populations. These modules dynamically modulate incoming inputs. The most significant biological effect occurs between LGN and layer IV neurons, leading to two variants:

- **LGN-only adaptation model:** applies synaptic depression only to LGN to layer IV connections.
- **Full adaptation model:** applies it to all synaptic connections.

Definition 8 (Synaptic Depression Module)**.** *Let L be a target layer, and Λ_L the set of input populations. For $L_{in} \in \Lambda_L$, let $x \in \mathbb{R}^{m_{in}}$ be its input at time t. Let $\sigma : \mathbb{R}^{m_{in}} \to \mathbb{R}^{m_{in}}$ be a joint RNN neuron module (as in Definition 3.3.2). Then the modulated input is:*

$$x' = \sigma(x)$$

Since these modules are recurrent, we again use TBPTT. This approach increases memory consumption substantially, especially when applied to all synaptic connections. To mitigate this, we:

- Use minimal GRU-based modules.
- Prefer the LGN-only variant to limit parameter growth.

Figure 3.2 depicts the complete model, including all additional modules discussed in this section.

In its current form, this extension is computationally demanding due to the all-to-all connectivity. However, we believe it holds significant potential for improving the model's ability to capture temporally dynamic behavior, especially when combined with future spatial connectivity constraints. In particular, we expect that this form of adaptability could help differentiate between response types, such as spontaneous activity and stimulus-driven responses. For instance, spontaneous responses should be less suppressed, while late responses to sustained stimuli may be attenuated due to short-term synaptic fatigue. Capturing this nuance may improve the biological realism and predictive accuracy of the model.

Chapter 4

Results

This chapter begins with a description of the experimental setup and key assumptions. We then give a brief overview of the dataset. The main section evaluates and compares different versions of our model, focusing on how each added component affects performance. Finally, we explore how the model's performance depends on the number of training experiments.

To simplify notation throughout this chapter, we use the following abbreviations for the model layers: X_ON and X_OFF represent the LGN ON and OFF cell populations. V1_Exc_L4 and V1_Inh_L4 refer to the excitatory and inhibitory cell populations in V1 layer IV, and V1_Exc_L23 and V1_Inh_L23 refer to those in V1 layer II/III. These match the layer names used in the SNN model code and will be used consistently throughout.

4.1 Experimental Setup and Technicalities

Unless otherwise specified, all experiments were run using the model setup and artificial dataset described in Chapter 3.

In addition to the setup from the previous chapter, we used several parameters chosen based on our experience during model development. These were mostly influenced by practical constraints such as limited GPU access, memory availability, or model-specific issues. Because thorough testing of all parameter combinations would have required too many resources, we selected values that we believed were reasonable. While we do not expect these parameters to drastically affect our findings, they could be optimized in future work.

ne important parameter is the batch size. As explained in Section 3.2, our dataset is made up of samples that each represent a single experiment. These had to be grouped into batches for training. We chose a batch size of 50 based on a balance of performance and hardware limits. Training RNNs on time-based data

is typically slow, since computations must happen in sequence and cannot be easily parallelized. Using larger batch sizes allows some parallel processing on GPUs, which helps speed things up. However, larger batches also require more memory.

Memory demands were particularly high when using certain RNN neuronal modules, described in Section 3.3.2, that rely on truncated backpropagation through time (TBPTT). TBPTT increases memory use significantly, especially when paired with small shared neural networks used in place of typical activation functions. To reduce this load, we merged time bins into 20 ms intervals, as described in Section 3.3.1, which helped but did not fully solve the issue. As noted in Section 3.2.4, we also had to limit our model to just 10% of the artificial neurons from each layer of the original SNN template to stay within memory limits. After weighing all these factors, we chose a batch size of 50, which worked reliably even with the most memory-intensive versions of our model, like the one using the synaptic depression module (Section 3.3.2).

Another training safeguard we used was gradient clipping, which prevents gradients from growing too large and causing numerical issues. In all our experiments, we clipped gradients to a maximum value of 10,000. This was mainly a precaution to avoid overflow errors during training, especially during early development phases when we were experimenting with activation functions other than LeakyTanh. This topic is discussed further in Section 3.3.1. In our final models, this gradient clipping had little to no effect on performance, but we kept it in place to ensure stability.

Finally, we want to mention the hardware used to run our experiments. Most were carried out on the Metacentrum computing cluster. While not every model variant needed a large GPU, we generally used GPUs with at least 40 GB of RAM. These are well-suited for deep learning tasks and were essential for models that relied on TBPTT. We also used machines with at least 8 CPU cores and 100 GB of RAM to support the computational load.

4.2 Dataset Overview

In this section, we present a statistical analysis of our dataset and evaluate the impact of the dataset simplifications we applied. First, we assess the effect of merging time bins from 1 ms to 20 ms, followed by an analysis of the influence of selecting a random subset comprising 10% of neurons. All scripts used for this analysis are provided in the supplementary materials and the project's GitHub repository.

Spike Count	1 ms	5 ms	10 ms	15 ms	20 ms
0	0.9944	0.9727	0.9491	0.9287	0.9105
1	0.0056	0.0266	0.0460	0.0598	0.0710
2	0.0000	0.0007	0.0046	0.0100	0.0147
3	0.0000	0.0000	0.0003	0.0013	0.0032
4	0.0000	0.0000	0.0000	0.0001	0.0005
5	0.0000	0.0000	0.0000	0.0000	0.0001

Table 4.1 Spike count distribution in the train dataset: This table shows the proportion of time bins containing 0 to 5 spikes for different time bin sizes. Spike counts above 5 are omitted as they occur rarely and have a negligible impact on the overall distribution.

4.2.1 Time Bin Merging Analysis

As described in Section 3.3.1, we merged the original 1 ms time bins into 20 ms intervals to accelerate computation and reduce data noisiness, while maintaining sufficient temporal resolution. We performed experiments using five bin sizes: 1 ms, 5 ms, 10 ms, 15 ms, and 20 ms. Due to the high computational cost of reprocessing the dataset for each binning size, we limited the analysis to this subset. Each configuration required substantial memory for processing and storage, which constrained the number of variants we could feasibly evaluate. The binning was introduced to significantly accelerate model training and reduce memory consumption.

Total Spike Counts Across Time Bins

We begin by comparing the distribution of spike counts across all time bins for each binning size. We hypothesize the following:

Claim 1 (Distribution of Spike Counts Across All Time Bins)**.** *The distribution of spike counts across time bins remains similar for bin sizes* $\{1, 5, 10, 15, 20\}$ *ms. This suggests that the temporal behavior of neuronal responses is largely preserved.*

Our assumption is that maintaining the binary-like properties of spike data should also preserve its temporal characteristics. Tables 4.1 and 4.2 show spike count distributions for the training and test datasets, respectively.

The distributions for both training and test sets are highly similar. Most time bins contain either zero or one spike, and only a small proportion include more than one. For the 20 ms bins, about 1.5% of time bins contain two or more spikes. Notably, the percentage of time bins with at least one spike increases from roughly 0.6% (1 ms bins) to 7% (20 ms bins). This reduction in sparsity supports the

Spike Count	1 ms	5 ms	10 ms	15 ms	20 ms
0	0.9944	0.9728	0.9493	0.9290	0.9107
1	0.0056	0.0265	0.0458	0.0597	0.0710
2	0.0000	0.0007	0.0045	0.0099	0.0146
3	0.0000	0.0000	0.0003	0.0013	0.0032
4	0.0000	0.0000	0.0000	0.0001	0.0005
5	0.0000	0.0000	0.0000	0.0000	0.0001

Table 4.2 Spike count distribution in the test dataset: This table shows the proportion of time bins containing 0 to 5 spikes for different time bin sizes. Spike counts above 5 are omitted as they occur rarely and have a negligible impact on the overall distribution.

hypothesis that the 20 ms bin size offers a reasonable balance between temporal resolution and data density.

Figures 4.1 and 4.2 further illustrate this by showing spike count distributions across all neuronal populations.

Spike Count Distribution Across Time

Next, we examine the distribution of spike counts across time for different bin sizes. We propose the following:

Claim 2 (Temporal Spike Count Distribution)**.** *The temporal distribution of spike counts per experiment remains consistent across all tested bin sizes. This supports the preservation of temporal response patterns.*

Figures 4.3 and 4.4 show the mean spike counts over time for all neuronal populations in the training and test datasets, respectively. These distributions were interpolated to the 1 ms time scale using cubic interpolation.

These plots show a clear reduction in noise with increasing bin size, particularly in excitatory layers. The training dataset appears noisier than the test set, likely due to the test set's smaller size and the averaging effect of multiple trials per experiment.

Importantly, the overall distribution shape is preserved across bin sizes. This indicates that binning effectively reduces noise without substantially altering temporal dynamics. Heatmaps in Figure 4.5 display Pearson correlation coefficients across different bin sizes, confirming strong similarity in spike distributions over time.

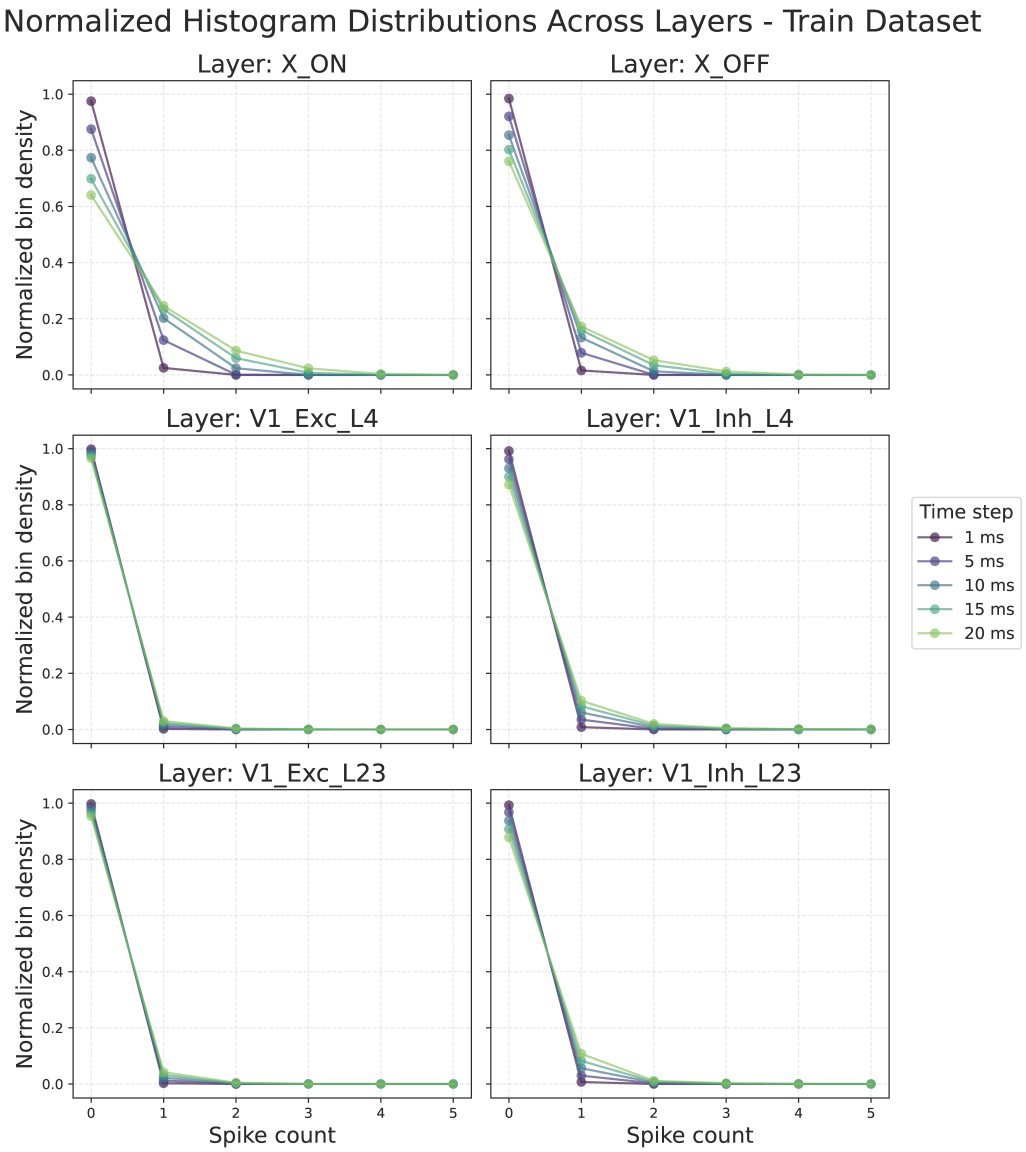

Figure 4.1 Evolution of spike counts per time bin with increasing bin size across all neuronal populations in the train dataset.

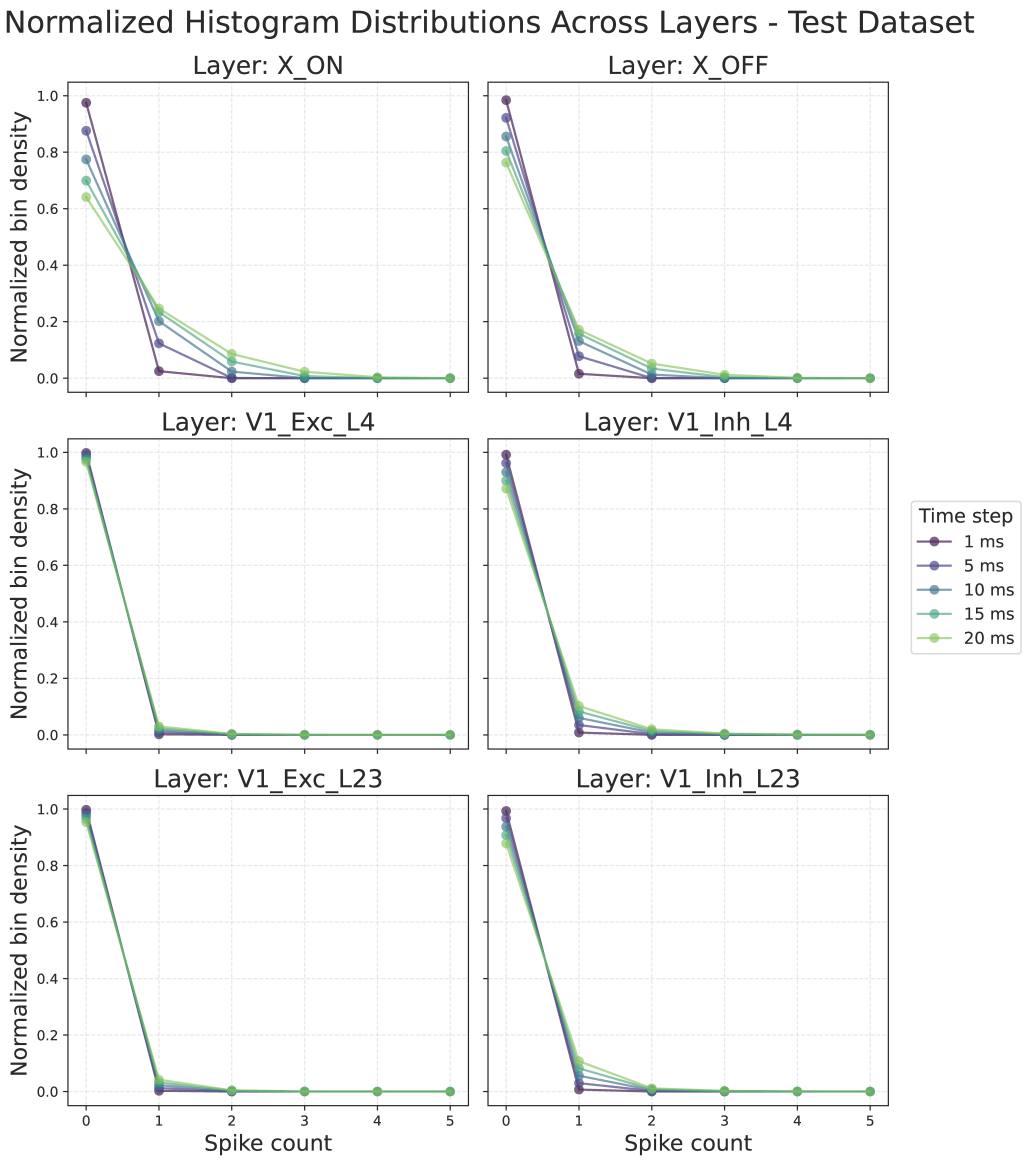

Figure 4.2 Evolution of spike counts per time bin with increasing bin size across all neuronal populations in the test dataset.

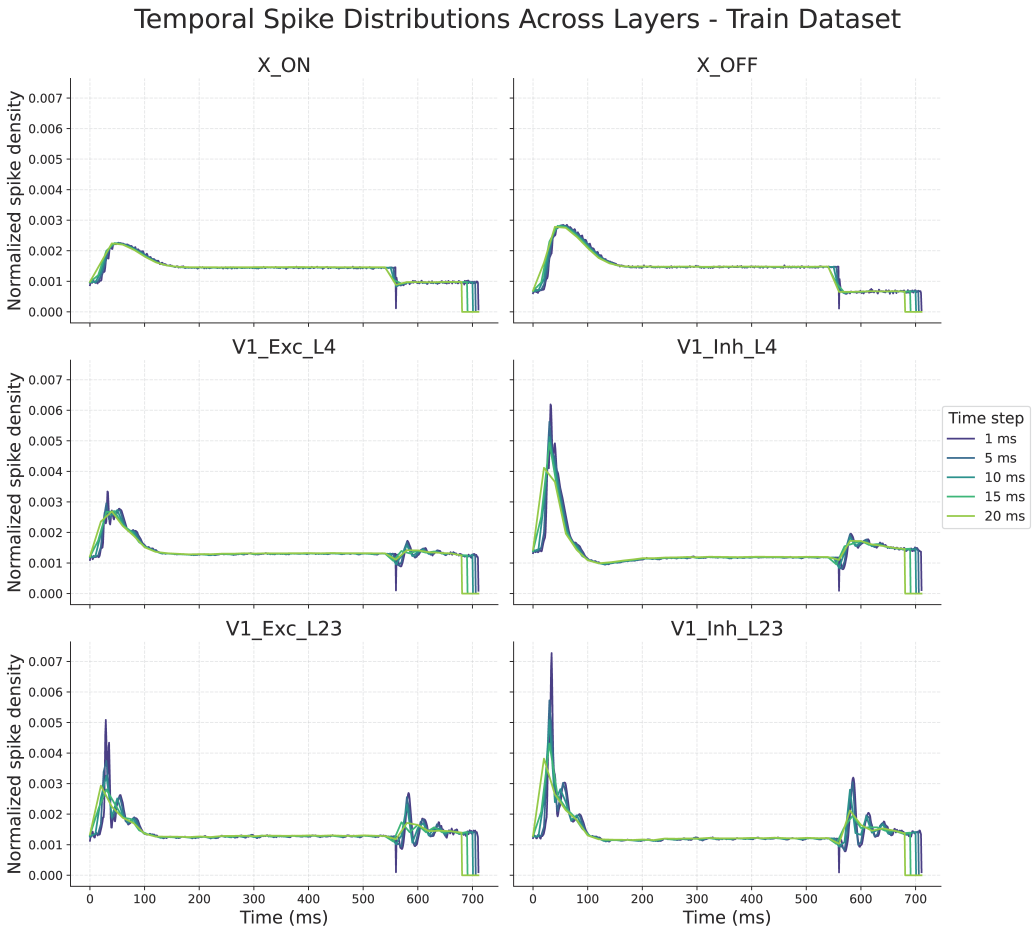

Figure 4.3 Comparison of temporal spike count distributions for different time bin sizes across all neuronal populations in the train dataset. The curves are interpolated to the original 1 ms resolution using cubic interpolation to improve line smoothness.

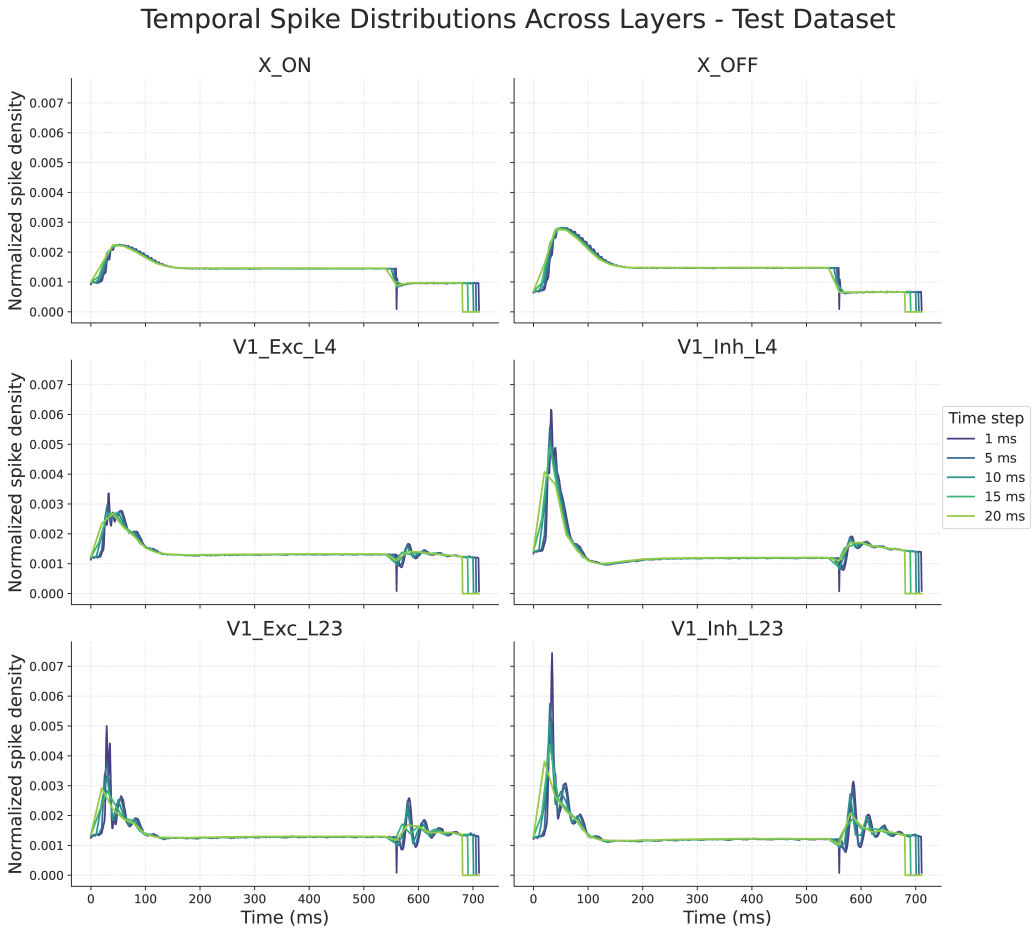

Figure 4.4 Comparison of temporal spike count distributions for different time bin sizes across all neuronal populations in the test dataset. The curves are interpolated to the original 1 ms resolution using cubic interpolation to improve line smoothness.

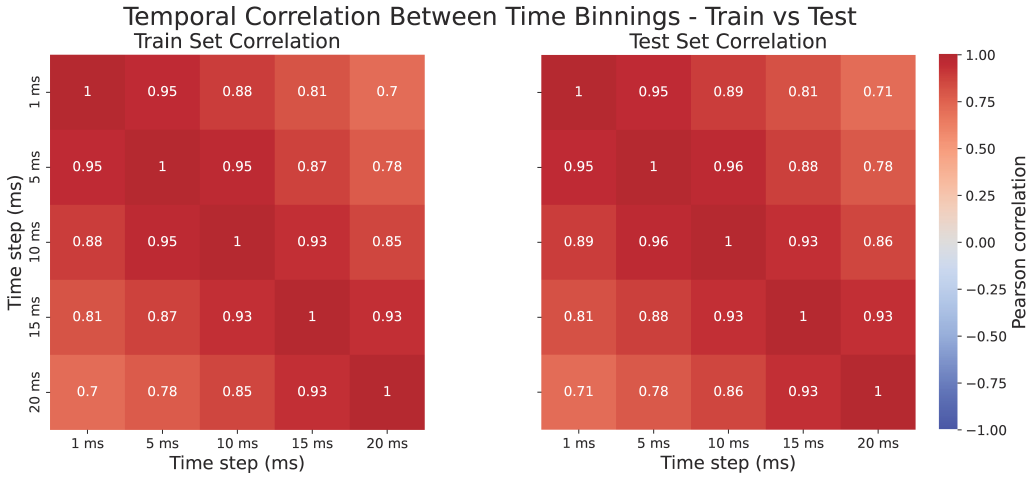

Figure 4.5 Heatmaps of Pearson correlation coefficients for spike count distributions over time between different time bin sizes in the train and test datasets.

Population Spike Count per Time Bin

Lastly, we assess how time binning influences the number of neurons spiking simultaneously within a single time bin. Throughout the analysis, we refer to this measure, the proportion of neurons active at each time step, as the *population spike count*. Population spike count provides insight into the collective temporal dynamics of neuronal populations and is widely studied in neuroscience (Singer [95]).

Our analysis focuses on the mean population spike count across all time steps. We state the following:

Claim 3 (Population Spike Count of Neuronal Populations Across Time Bins)**.** *The mean population spike count of neuronal populations remains consistent across different time bin sizes. This implies that temporal structure of the dataset is largely preserved.*

Figures 4.6 and 4.7 show population spike count distributions for each layer in the training and test datasets. Tables 4.3 and 4.4 summarize the mean and variance of population spike count values.

Population spike count increases with larger time bins, reflecting a higher likelihood of coincident spiking. This effect is most pronounced in LGN layers, especially X_ON, while V1 excitatory layers show a milder change.

Based on these findings, we conclude that a binning size of 20 ms influences the temporal dynamics of neuronal responses and may consequently affect the resulting neuronal predictions. However, due to the computational complexity of our task, it was not feasible to develop and train models using a smaller time bin

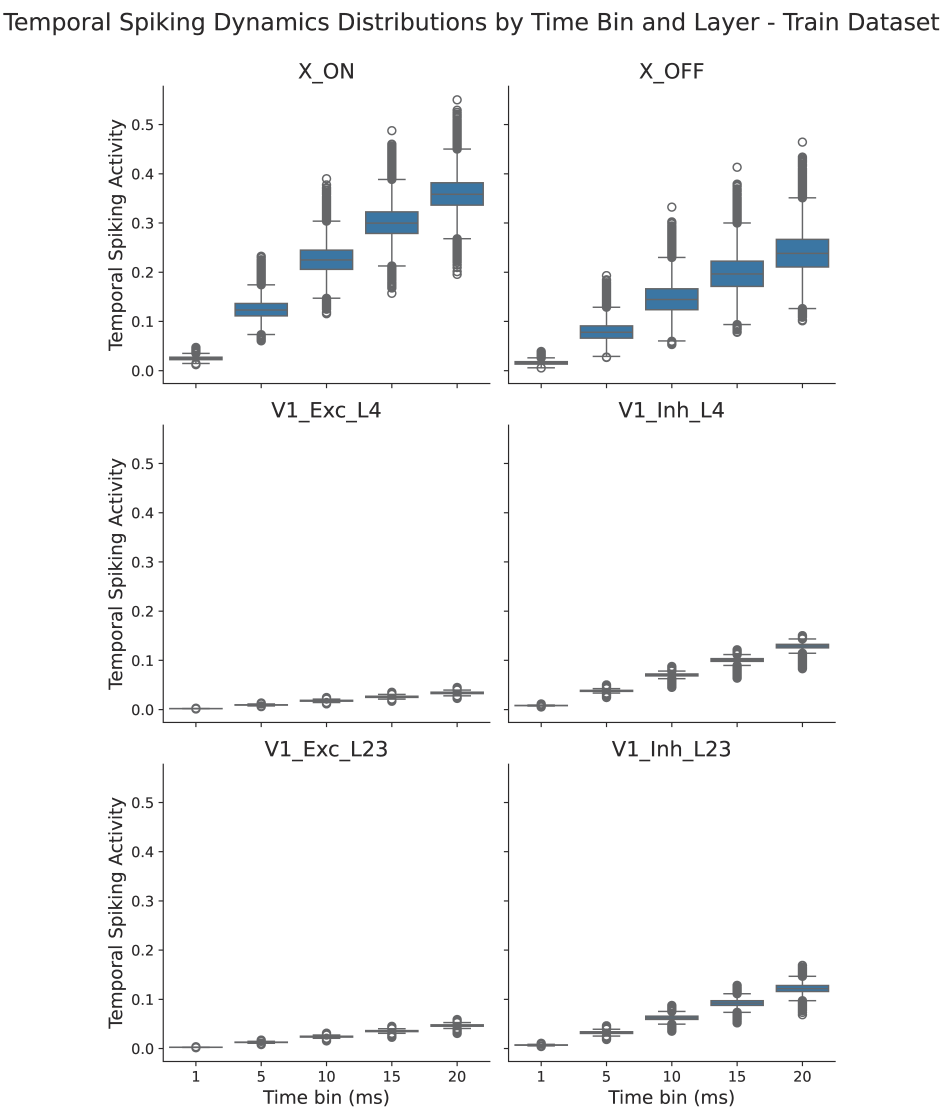

Figure 4.6 Distribution of mean population spike count across different time bin sizes for all neuronal populations in the train dataset. The boxplot represents the distribution of mean population spike count values calculated across all experiments

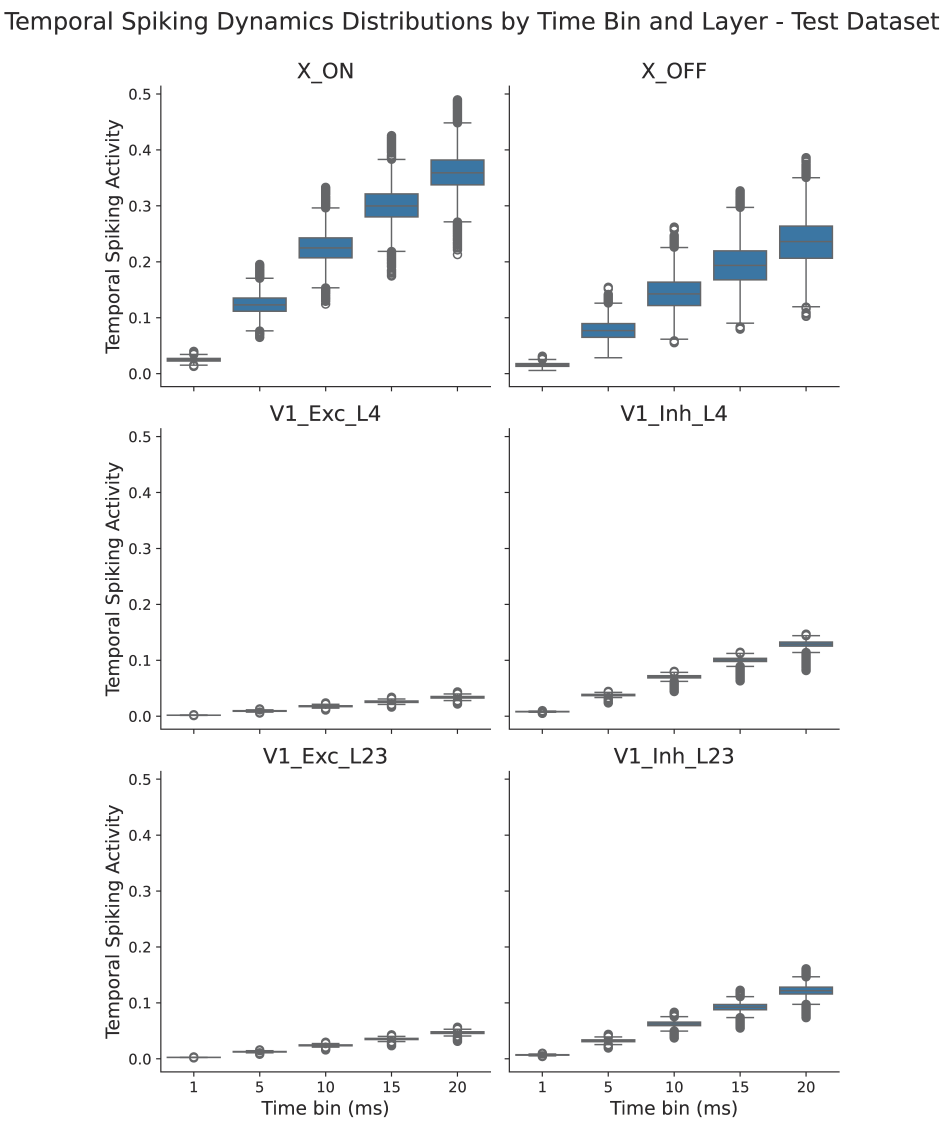

Figure 4.7 Distribution of mean population spike count across different time bin sizes for all neuronal populations in the test dataset. The boxplot represents the distribution of mean population spike count values calculated across all experiments

time step	mean	variance
1	0.0101	0.0001
5	0.0494	0.0018
10	0.0912	0.0057
15	0.1256	0.0098
20	0.1551	0.0134

Table 4.3 Summary of population spike count statistics across time bin sizes in the train dataset: This table reports the mean and variance of population spike count across all layers for each time bin size.

time step	mean	variance
1	0.0101	0.0001
5	0.0491	0.0017
10	0.0908	0.0057
15	0.1250	0.0097
20	0.1545	0.0133

Table 4.4 Summary of population spike count statistics across time bin sizes in the test dataset: This table reports the mean and variance of population spike count across all layers for each time bin size.

size. Therefore, this limitation must be carefully considered during both model development and the interpretation of results.

4.2.2 Model Subset Selection Analysis

In this section, we analyze the impact of selecting a subset of neurons from the original SNN model. As discussed in Section 3.2.4, we selected only 10% of the neurons due to memory constraints and the computational demands of model training. Our primary interest lies in assessing how this subset selection affects the temporal properties of the dataset, which are central to our research.

All experiments in this section are conducted on the dataset with a 20 ms time bin size (as used in our model) and focus on the training dataset unless stated otherwise. As shown previously in Section 3.3.1, the results from the training and test datasets are largely consistent.

Total Spike Counts Across Time Bins

We begin by analyzing the distribution of spike counts in the time bins for the full dataset and for the subset datasets. Table 4.5 summarizes the mean spike count ratios across all model subsets compared to the full dataset.

Spike Count	Full Dataset Ratio	Subsets Mean Ratio	Subsets Standard Deviation
0	0.9105	0.9102	0.0004
1	0.0710	0.0712	0.0003
2	0.0147	0.0148	0.0001
3	0.0032	0.0032	0.0000
4	0.0005	0.0005	0.0000
5	0.0001	0.0001	0.0000

Table 4.5 Comparison of spike count distributions between full and subset datasets: This table presents the mean spike count ratios and standard deviations across all model subsets, relative to the full dataset.

The table shows that the differences in spike count distributions between the full and subset datasets are minimal, with low standard deviations. This suggests that randomly selecting a subset of neurons does not significantly impact the overall spike count distribution.

Spike Count Distribution Across Time

Next, we compare the temporal spike count distributions of the subset datasets with that of the full dataset. Figure 4.8 shows the mean temporal behavior across layers for both the full dataset and the average of all subsets. The shaded area represents the standard deviation across subsets.

The plot demonstrates that the mean temporal patterns of the subsets closely match those of the full dataset. The small standard deviation further confirms that the temporal properties are preserved despite the neuron subset selection.

Population spike count of Neuronal Populations in Subsets

Finally, we assess the effect of subset selection on neuronal population spike count. Since we are using only a portion of the neurons from the full model, it is possible that groups of neurons that typically spike together may be disrupted, potentially affecting population spike count.

Figure 4.9 displays a boxplot and jitter plot comparing the population spike count values of the full dataset and all subsets.

As shown, the population spike count distributions are largely similar, particularly in the excitatory layers. This aligns with earlier findings that population spike count variance was low across time bins. Although the jitter plot suggests a slightly narrower spread in the LGN layers for the subsets, the boxplots do not indicate a significant difference.

Based on these observations, we conclude that random selection of 10% of neurons from the full SNN model does not substantially affect the dataset's

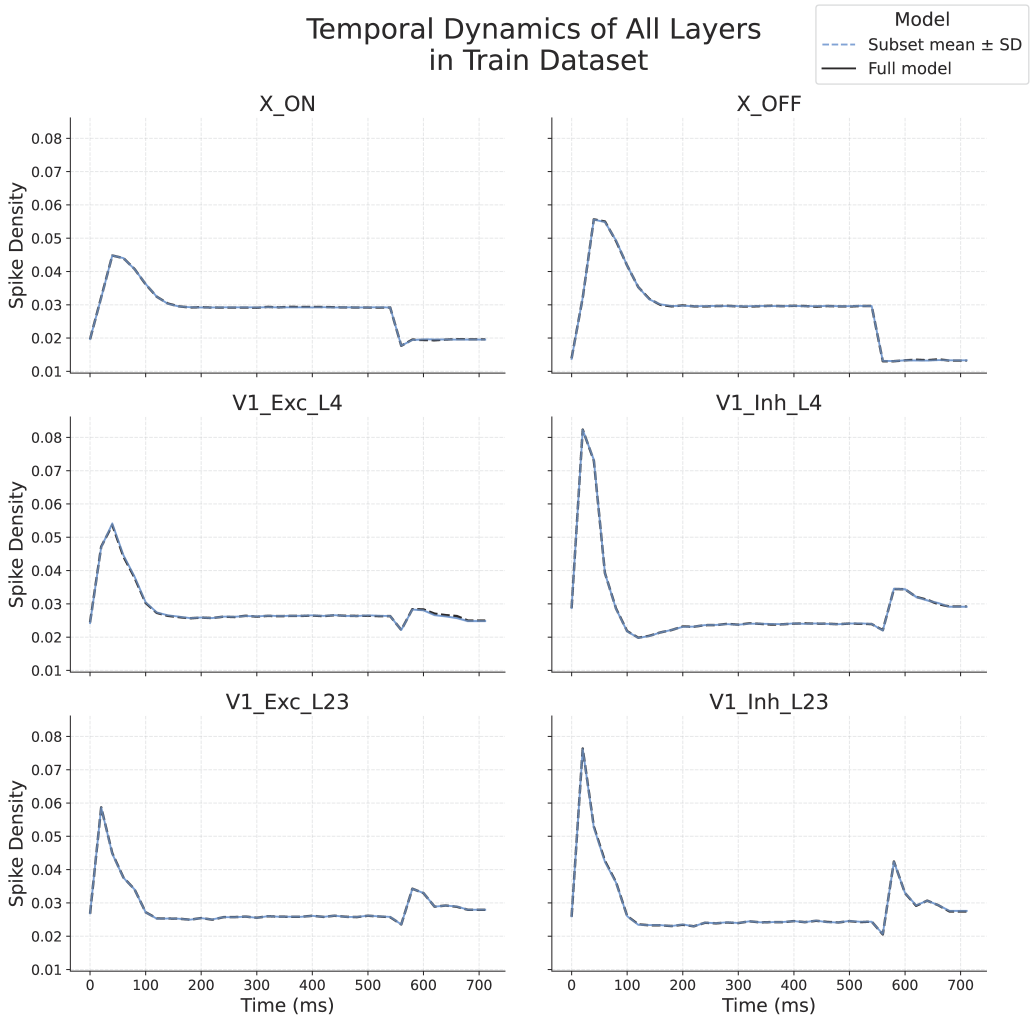

Figure 4.8 Comparison of mean temporal activity across layers for the full dataset and the average of all subsets. The shaded area indicates the standard deviation across subsets, which is very small and therefore barely visible in the plot.

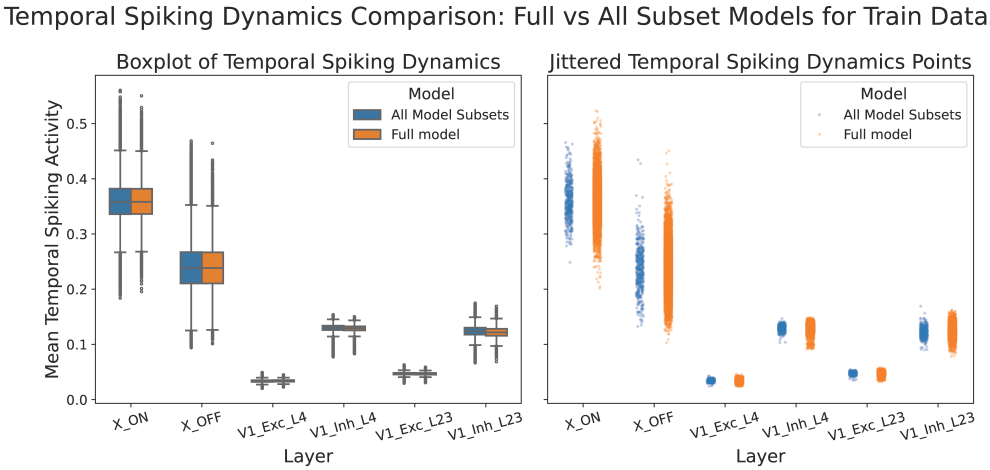

Figure 4.9 Boxplot and jitter plot comparing population population spike count values between the full dataset and all subsets. Each data point represents the mean population spike count of one experiment.

statistical or temporal properties. Nevertheless, slight changes in population spike count, particularly in LGN layers, should be taken into account when interpreting model performance results.

4.3 Model Evaluation

This section evaluates the performance of our models by comparing different versions and analyzing how each additional module influences the results. We start with a high-level comparison of all model variants, then examine individual performances, and finally discuss the limitations and possible causes of prediction errors.

Unless noted otherwise, all results are aggregated across 20 randomly selected neuronal sub-populations, each representing 10% of the original dataset. This selection method, described in Section 3.2.4, helps reduce bias and better reflect overall model behavior.

We follow a consistent naming convention for the models, primarily based on definitions in Section 3.3. However, to distinguish between certain configurations, we introduce some new labels:

- **simple (tanh)**: A simple model with a tanh activation function (Section 3.3.1).
- **simple (leakytanh)**: A simple model using a leakytanh activation function (Section 3.3.1).

Model Variant	Epochs	lr	n-ls	n-nl	n-res	s-ls	s-nl	n-tbptt
simple (tanh)	10	0.000008	-	-	-	-	-	-
simple (leakytanh)	10	0.000075	-	-	-	-	-	-
dnn joint	10	0.000010	10	5	True	-	-	-
dnn separate	10	0.000010	10	5	True	-	-	-
rnn (steps 5)	40	0.000030	10	3	True	-	-	5
rnn (steps 10)	40	0.000030	10	3	True	-	-	10
syn adapt lgn (steps 5)	40	0.000030	10	3	True	10	2	5

Table 4.6 Model Setup for Evaluation: Configuration of each model variant used during evaluation. "Model Variant" indicates the model type. Other columns: "Epochs" - number of training epochs, "lr" - learning rate, "n-ls" - neuron module layer size, "n-nl" - number of layers in the neuron module, "n-res" - whether residual connections are used in the neuron module, "s-ls" - synaptic depression module layer size, "s-nl" - number of layers in the synaptic depression module, "n-tbptt" - number of TBPTT steps. A dash ("-") means the value is not applicable for that model.

- **dnn joint**: A model using feed-forward joint neuron modules (Section 3.3.2), where the input processing does not differentiate between excitatory and inhibitory neuronal populations.
- **dnn separate**: A model using feed-forward separate neuron modules (Section 3.3.2), where the input processing differentiates between excitatory and inhibitory neuronal populations.
- **rnn (5 steps)**: A model using RNN neuron modules with 5 TBPTT steps (Section 3.3.2).
- **rnn (10 steps)**: A model using RNN neuron modules with 10 TBPTT steps.
- **syn adapt lgn (5 steps)**: A model using RNN neuron modules also with synaptic depression modules applied only to LGN connections, using 5 TBPTT steps (Section 3.3.2). The term *adapt* refers to the naming in our implementation of the module.

All models were trained and evaluated using the general setup outlined in Section 4.1. Specific hyperparameters for each model are listed in Table 4.6. These parameters were primarily selected through grid search, supplemented by empirical tuning based on development experience. More details on this process will follow in subsequent sections.

Model Variant	N-CC (mean)	P-CC (mean)	N-CC (std)	P-CC (std)
rnn (10 steps)	0.9212	0.7500	0.0084	0.0082
rnn (5 steps)	0.9176	0.7471	0.0103	0.0089
syn adapt lgn (5 steps)	0.8935	0.7275	0.0043	0.0042
dnn joint	0.8803	0.7168	0.0021	0.0034
simple (leakytanh)	0.8778	0.7147	0.0014	0.0032
dnn separate	0.8430	0.6864	0.0940	0.0766
simple (tanh)	0.2767	0.2252	0.0400	0.0321

Table 4.7 Summary of Model Performance Metrics: Mean and standard deviation of normalized and Pearson correlation coefficients for each model variant, calculated across all model subset variants. Results are sorted by the mean normalized cross-correlation. "Model Variant" indicates the model type. Other columns: "N-CC (mean)" - mean normalized CC, "P-CC (mean)" - mean pearson CC, "N-CC (std)" - standard deviation normalized CC, "P-CC (std)" - standard deviation pearson CC.

4.3.1 Comparative Evaluation of Model Types

This section provides a comparative analysis of all model variants using global performance metrics.

We begin by evaluating each model's performance using Pearson correlation coefficients and normalized cross-correlations, as detailed in Sections 2.3.1 and 2.3.2. Table 4.7 presents the summary statistics for each model, while Figure 4.10 visualizes the distribution of Pearson and normalized cross-correlation values across all model subset variants.

The results indicate that Pearson and normalized cross-correlation metrics exhibit similar behavior, with the primary difference being a consistent offset due to the normalization process. Given this similarity, we will primarily use normalized cross-correlation in subsequent analyses unless otherwise noted.

One clear outlier in performance is the simple (tanh) model, which significantly underperforms compared to all other models. In contrast, the simple (leakytanh) model, utilizing out custom task-specific activation function, performs comparably to the dnn joint and dnn separate models. This indicates that a well-designed activation function can approximate the functionality of more complex feed-forward modules.

The RNN-based models (rnn (5 steps), rnn (10 steps) and syn adapt lgn (5 steps)) demonstrate superior performance, highlighting the importance of incorporating memory through recurrent structures and TBPTT training. This result suggests that temporal dependencies in neuronal activity are better captured by models with memory components.

Another potential reason why the simple and feed-forward models perform reasonably well could be their ability to capture the mean spiking pattern of the

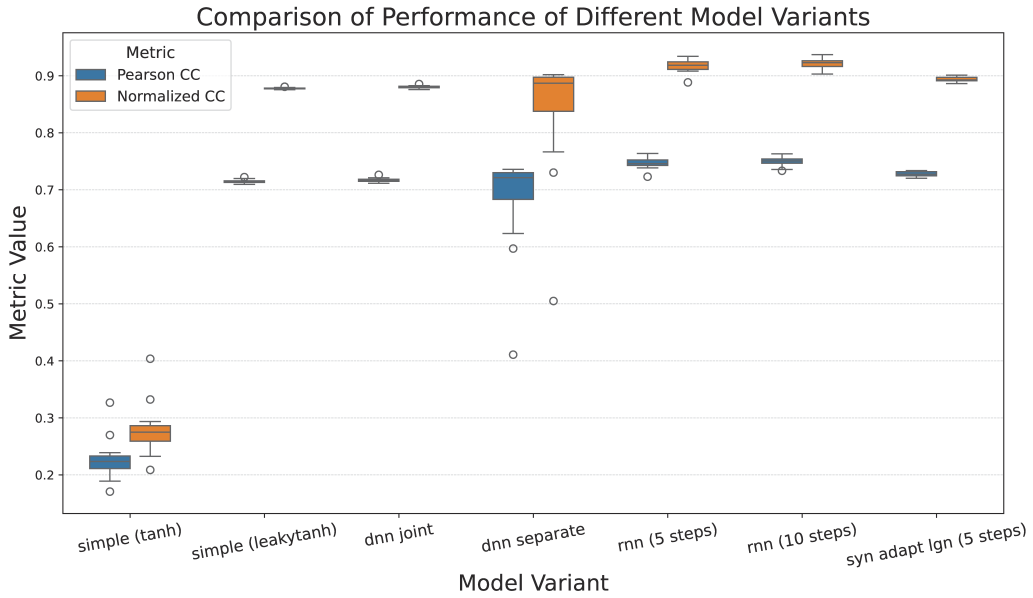

Figure 4.10 Boxplot comparing the distribution of Pearson and normalized cross-correlation values across all model variants.

neuronal populations. As previously discussed in Section 2.2.2. The predicted responses are relatively sparse, and high correlation values may partly reflect this sparsity rather than true predictive accuracy. In contrast, the improved performance observed in TBPTT-based models may reflect their enhanced ability to model temporal dynamics in neuronal activity.

Interestingly, the dnn separate model exhibits a high degree of variability of correlation coefficients across subsets, which is unexpected. Although designed to separate excitatory and inhibitory processing and initially showed results similar to dnn joint during development, its inconsistency may be due to the limited number of subsets (20), or it could reflect limitations in the model architecture when dealing with separate pathways. Additional experiments with a larger number of subsets would be needed to draw firmer conclusions.

The top-performing model overall is rnn (10 steps), which achieves the highest mean normalized correlation and maintains consistent performance across subsets. Increasing the number of TBPTT steps appears beneficial, though further increases were limited by computational constraints.

An unexpected observation is that the syn adapt lgn (5 steps) model underperforms compared to rnn neuron module variants without synaptic depression. This may be due to suboptimal hyperparameter tuning and insufficient number of TBPTT steps, which were not fully explored due to resource limitations.

Finally, we analyze the population spike count patterns of model predictions

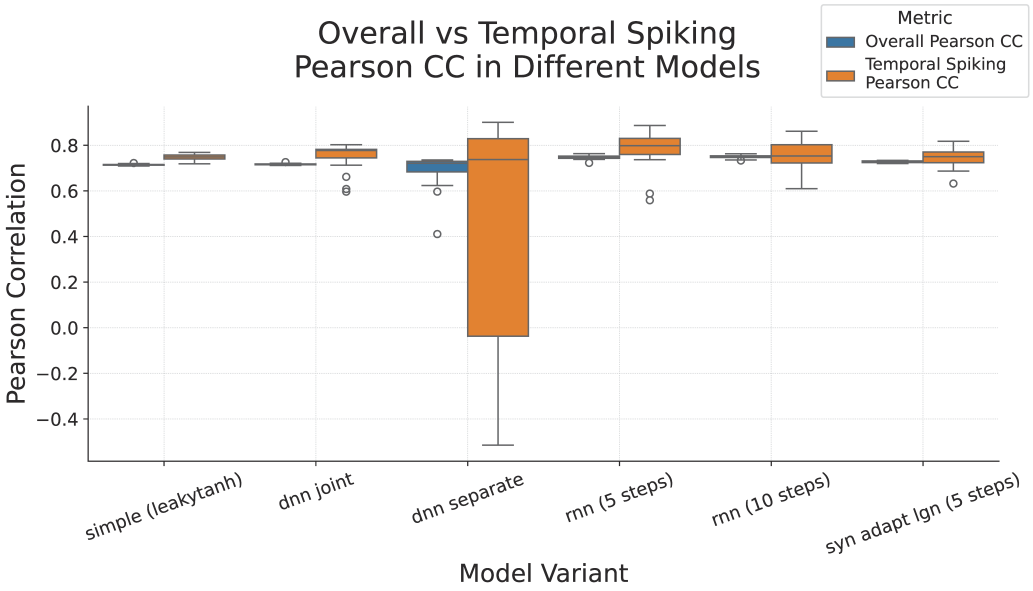

Figure 4.11 Boxplot comparing Pearson correlation coefficients for model predictions and temporal spiking dynamics curves across all layers and model subsets.

by comparing their *temporal spiking dynamics curves*, which represent the fraction of simultaneously spiking neurons at each time step (see Section 4.2.1). Figure 4.11 shows the distribution of Pearson correlation coefficients for both overall predictions and temporal spiking dynamics curves. Note that simple (tanh) is excluded due to technical issues during evaluation.

The variance in population spike count correlations is higher than in overall predictions for all models, particularly those trained with TBPTT. We hypothesize it is because these models better capture temporal dynamics. Notably, rnn (5 steps) slightly outperforms rnn (10 steps) in this metric, although this may be an artifact of limited subset size.

The dnn separate model again displays irregular behavior with wide variability in population spike count correlation. This could be due to its architecture, which might struggle with modeling complex input dynamics without memory, or due to technical issues that were not detected. Additional investigation with a more comprehensive dataset is necessary to confirm these findings.

4.3.2 Temporal Dynamics of Predicted vs. Actual Neural Activity

This section focuses on evaluating the temporal behavior of the model predictions. Specifically, we examine the temporal spiking dynamics curves of individual

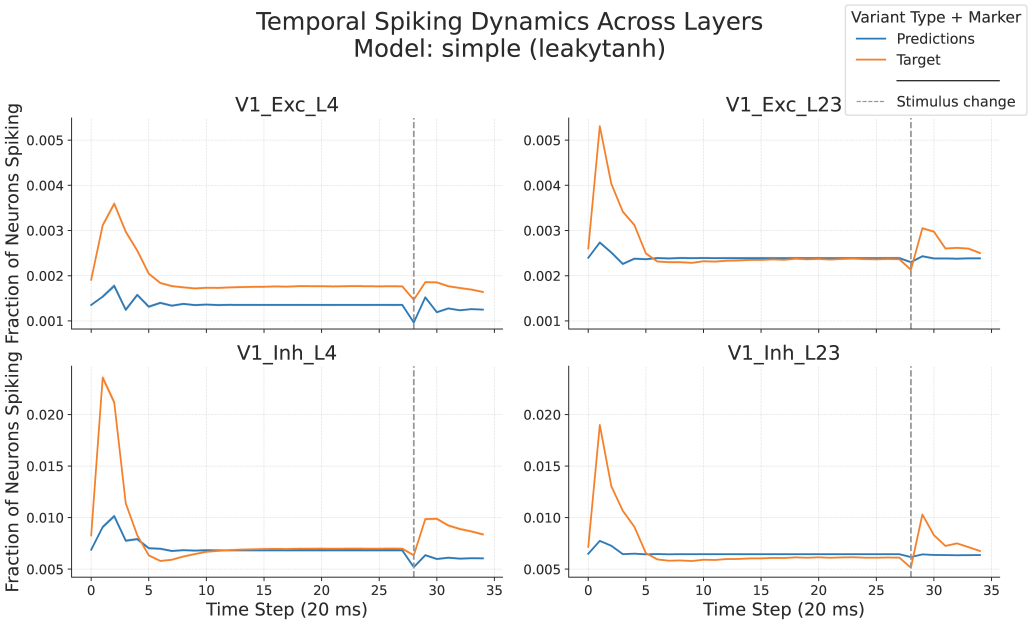

Figure 4.12 Mean temporal spiking dynamics curve of the simple (leakytanh) model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

model variants and compare them with the corresponding target temporal spiking dynamics curves. Due to unresolved technical issues during evaluation, the simple (tanh) model is excluded from this analysis, as its predictions could not be retrieved. However, given that the simple (tanh) model has already been shown to perform significantly worse than other models, its exclusion is not expected to compromise the validity of our conclusions.

The primary motivation behind analyzing temporal spiking dynamics curves is to assess whether the models can capture temporal patterns in neuronal activity, an essential aspect of biological realism. Accurate modeling of such dynamics would represent a meaningful advancement over traditional neural network approaches to system identification, as previously discussed in Section 2.2.2.

Simple Model

Figure 4.12 shows the mean temporal spiking dynamics curve for the simple (leakytanh) model, alongside the target temporal spiking dynamics curve, aggregated across all trials and model subset variants.

The temporal spiking dynamics curve for the simple (leakytanh) model exhibits relatively flat behavior across time, suggesting that the model is primarily

capturing the mean spiking rate rather than dynamic temporal fluctuations. This behavior results in reasonably accurate predictions during the later phase of stimulus presentation but fails to account for the initial spike in activity during the onset of natural stimuli.

Interestingly, the model successfully captures the decline in population spike count following the switch to blank stimuli, performing reasonably well in comparison to more complex models in this regard.

We hypothesize that the relatively strong performance in capturing average activity levels is due to the use of the specially designed leakytanh activation function. This function constrains predictions within a biologically plausible range, encouraging outputs that align with the mean neuronal response. This interpretation is further supported by the poor performance of the simple (tanh) model, which lacks this task-specific regularization. Although temporal spiking dynamics curve data for the tanh variant are unavailable, its previously reported low correlation values suggest inferior temporal modeling compared to the leakytanh version.

Feed-Forward Neuronal Module Variants

In this section, we examine the temporal spiking dynamics curves of the dnn joint and dnn separate models. Figure 4.13 presents the results for the dnn joint model, while Figure 4.14 displays those for the dnn separate model.

As previously noted, the dnn separate model exhibits unusually high variability, which is also reflected in the large error margins of its temporal spiking dynamics curve. Since potential explanations for this behavior have already been discussed, we will not revisit this model in detail here.

In contrast, the dnn joint model shows modest improvement in capturing early-stage spiking activity in response to natural stimuli compared to the simple (leakytanh) model. However, the overall pattern remains relatively flat, suggesting that the model continues to prioritize mean spiking activity over dynamic temporal features.

These results suggest that while feed-forward neuron modules may offer a slight enhancement in temporal representation, their lack of internal memory limits their ability to model time-dependent behavior. The observed performance gain is likely attributable to the increased representational capacity of standard feed-forward networks over a fixed activation function like leakytanh. Nonetheless, without mechanisms to retain information about past stimuli, these architectures fall short in accurately modeling temporal dynamics in neuronal activity.

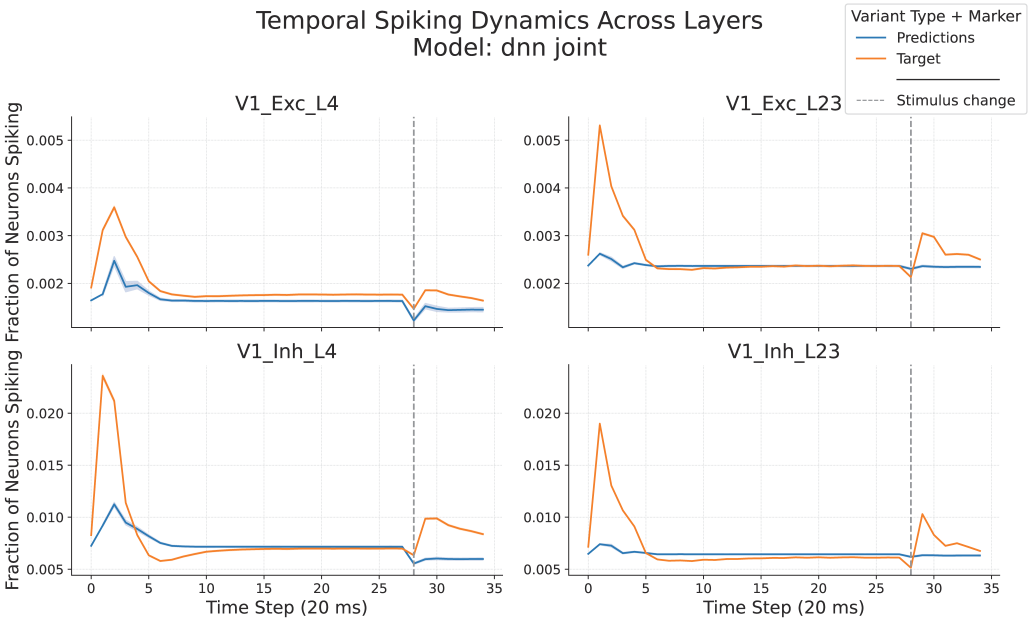

Figure 4.13 Mean temporal spiking dynamics curve of the dnn joint model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

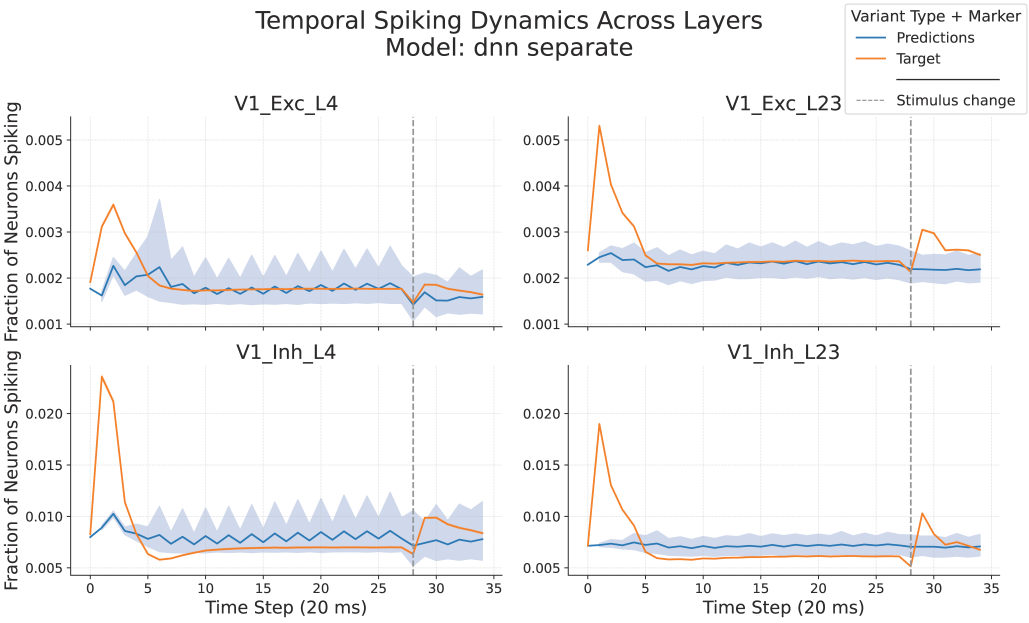

Figure 4.14 Mean temporal spiking dynamics curve of the dnn separate model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

RNN Neuronal Modules Variants

TBPTT enables the RNN model to better account for temporal dependencies by allowing gradients to propagate through a defined number of time steps, thereby supporting more accurate modeling of dynamic behaviors in neuronal activity.

Although earlier models incorporated recurrent connections to enhance biological plausibility and support temporal self-modulation, they were trained exclusively using a *teacher-forcing* strategy (Lamb et al. [96]). In contrast, the RNN variants evaluated in this section preserve hidden states across multiple time steps by utilizing both the teacher-forcing strategy and TBPTT, allowing the model to learn directly from the temporal structure during training.

Figures 4.15 and 4.16 display the temporal spiking dynamics curves for models using RNN neuron modules trained with TBPTT over 5 and 10 time steps, respectively. Both models demonstrate an improvement in capturing temporal dynamics, particularly in the early phase of natural stimulus presentation. This phase is characterized by heightened spiking activity, which the TBPTT models successfully reproduce.

Despite these improvements, a notable limitation remains. The models struggle to accurately capture neuronal responses to blank stimuli. In the model predictions, the population spike count following the transition to the blank phase is comparable to that observed during later stages of the natural stimulus. However, in the target responses, the spiking activity during the blank phase, often referred to as spontaneous neuronal activity (see Section 3.2.1), is notably higher than during prolonged natural stimulus exposure. This discrepancy suggests that the models do not fully account for the suppression of neuronal responses associated with long-term stimulus presentation.

A comparison of the two model variants suggests that increasing the number of TBPTT time steps leads to enhanced model performance. The rnn (10 steps) model not only exhibits more consistent predictions across subset variants but also achieves the highest correlation scores among all tested models, both in normalized and Pearson correlation coefficients.

When reviewing the population spike count-based Pearson correlation coefficients (Figure 4.11), we observe increased variability in TBPTT-trained models relative to their non-TBPTT counterparts. This may be attributed to the greater dynamic range captured by these models. While non-TBPTT models tend to emphasize the stable mean activity, TBPTT models more effectively capture transient dynamics, albeit with slightly less accuracy in static phases.

These findings indicate that models based on RNN neuronal modules trained with TBPTT represent a significant step toward biologically plausible neural network models that can robustly capture temporal patterns. As discussed in Sections 2.2.2 and 2.2.3, the ability to model dynamic neuronal activity is a key

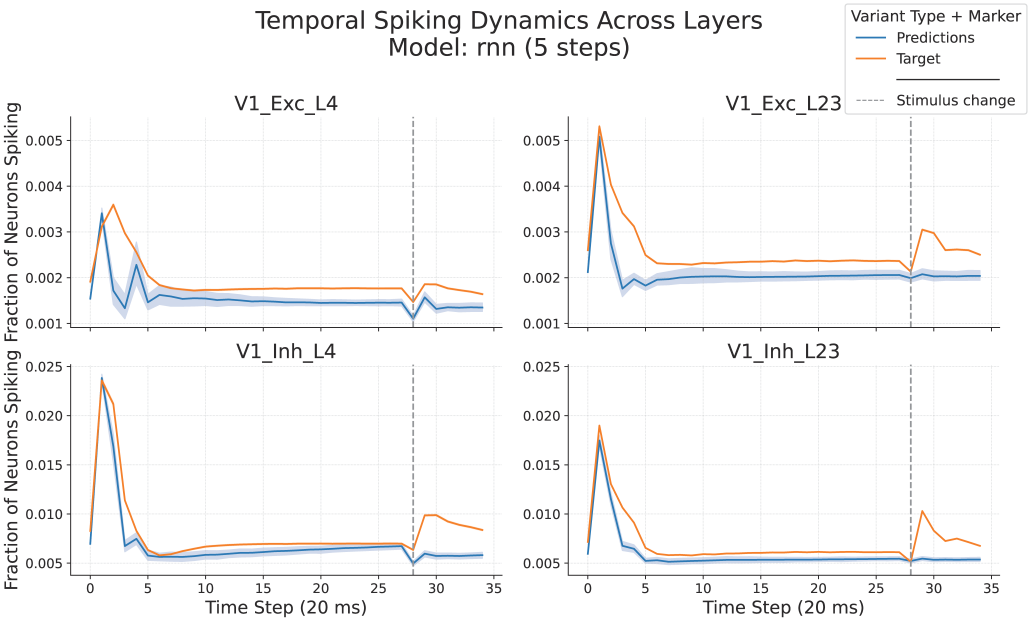

Figure 4.15 Mean temporal spiking dynamics curve of the rnn (5 steps) model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

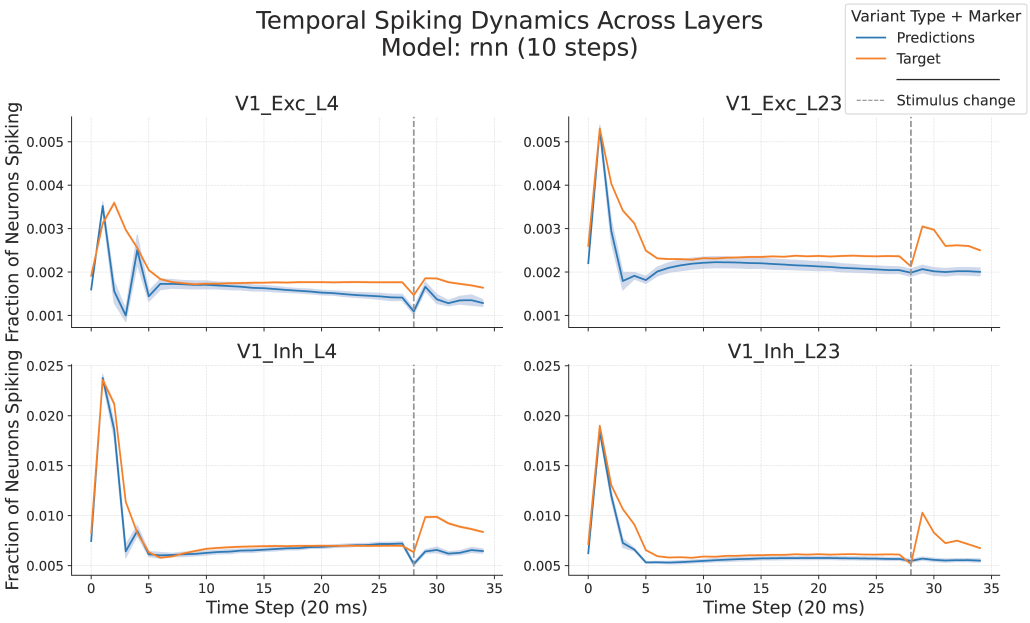

Figure 4.16 Mean temporal spiking dynamics curve of the rnn (10 steps) model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

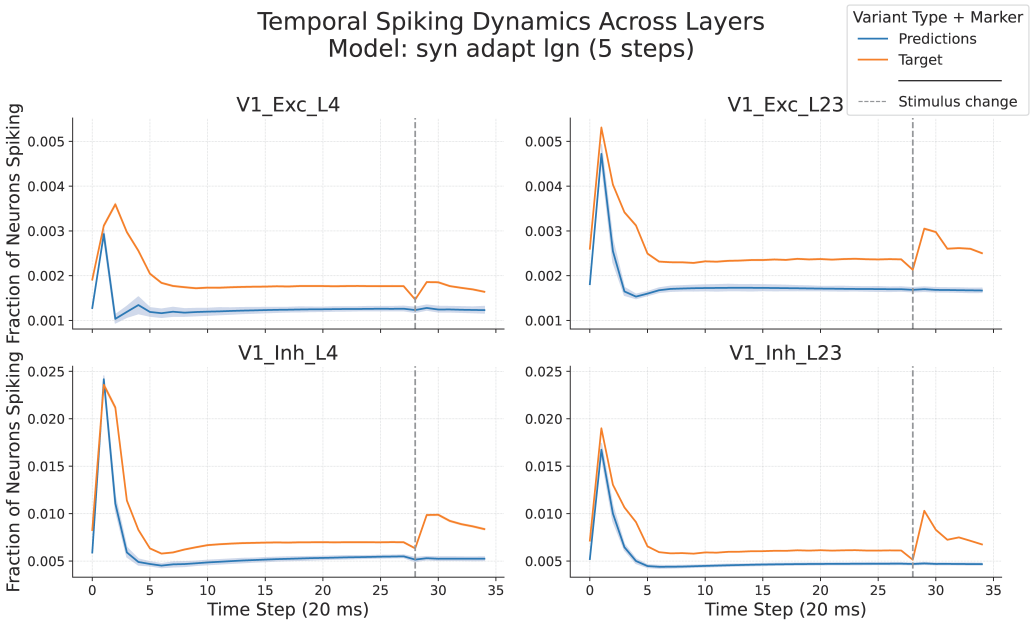

Figure 4.17 Mean temporal spiking dynamics curve of the syn adapt lgn (5 steps) model predictions compared to the target temporal spiking dynamics curve. Results are averaged over all trials and model subsets. Error bars represent standard deviation. The dashed vertical line marks the transition from natural to blank stimuli.

criterion for developing interpretable and biologically grounded deep learning systems.

Synaptic Depression Model

In the final part of our temporal spiking dynamics curve analysis, we turn to the evaluation of a model incorporating a synaptic depression module. As discussed previously, the temporal dynamics of spontaneous neuronal responses during the blank stimulus phase remain inadequately captured by earlier models. This motivates the integration of synaptic depression mechanisms, which aim to more accurately model the adaptive response of an individual synapses and improve temporal fidelity.

Synaptic depression experiments were performed on a variant model utilizing five TBPTT time steps, with the depression mechanism applied specifically to LGN connections. This choice reflects biological evidence indicating that LGN synapses are particularly influenced by synaptic depression. The resulting temporal spiking dynamics curve is shown in Figure 4.17.

This experimental configuration was chosen due to technical constraints. The inclusion of synaptic depression in RNNs dramatically increases memory require-

ments, especially with longer TBPTT sequences, necessitating high-memory GPU resources. Additionally, training times were significantly extended. As a result, a comprehensive grid search to optimize hyperparameters was not feasible. The hyperparameters used for this model were selected empirically, without the level of tuning applied to other model variants.

The temporal spiking dynamics curve shows that, while the model performs better than non-TBPTT variants, it underperforms relative to RNN models without synaptic depression. Notably, performance appears degraded across the entire stimulus timeline. There is reduced accuracy in modeling the transition from natural to blank stimuli and diminished representation of average neuronal responses in later stages. Furthermore, the model fails to improve prediction of spontaneous responses during the blank phase.

These findings are unexpected, as the incorporation of biologically inspired mechanisms was anticipated to enhance performance. We hypothesize that the underwhelming results are due to the suboptimal hyperparameter configuration and the limited temporal depth (5 TBPTT steps). Additionally, the training dataset may lack sufficient variability to enable the model to generalize dynamic patterns effectively. Further experimentation with improved computational resources and a more diverse dataset may be necessary to fully evaluate the potential of synaptic depression mechanisms in temporal modeling.

4.3.3 Statistical Comparison of Model Performance

To further support our conclusions regarding model performance, we conducted a pairwise one-sided Mann-Whitney U test [97]. This non-parametric test was chosen because the underlying distribution of normalized cross-correlation (CC) values for each model is unknown, making parametric tests less appropriate. The one-sided Mann-Whitney U test evaluates whether the distribution of one sample is statistically greater than or equal to another, without assuming a specific distributional form.

We applied this test to all pairwise combinations of model variants, comparing their normalized CC distributions. The results are visualized as a heatmap in Figure 4.18, where each cell represents the p-value of a one-sided test. In this visualization, a row model (”Model A”) is tested against a column model (”Model B”). A p-value below the significance level $\alpha = 0.05$ indicates that ”Model A” significantly outperforms ”Model B” in terms of normalized CC.

The statistical results largely corroborate the insights obtained from previous visual analyses of mean, variance and population spike count. At the $\alpha = 0.05$ level, we reject the hypothesis that the simple (tanh) model performs better than any other variant, reinforcing its role as the weakest model. Among the more complex models, the results generally support the idea that additional architectural

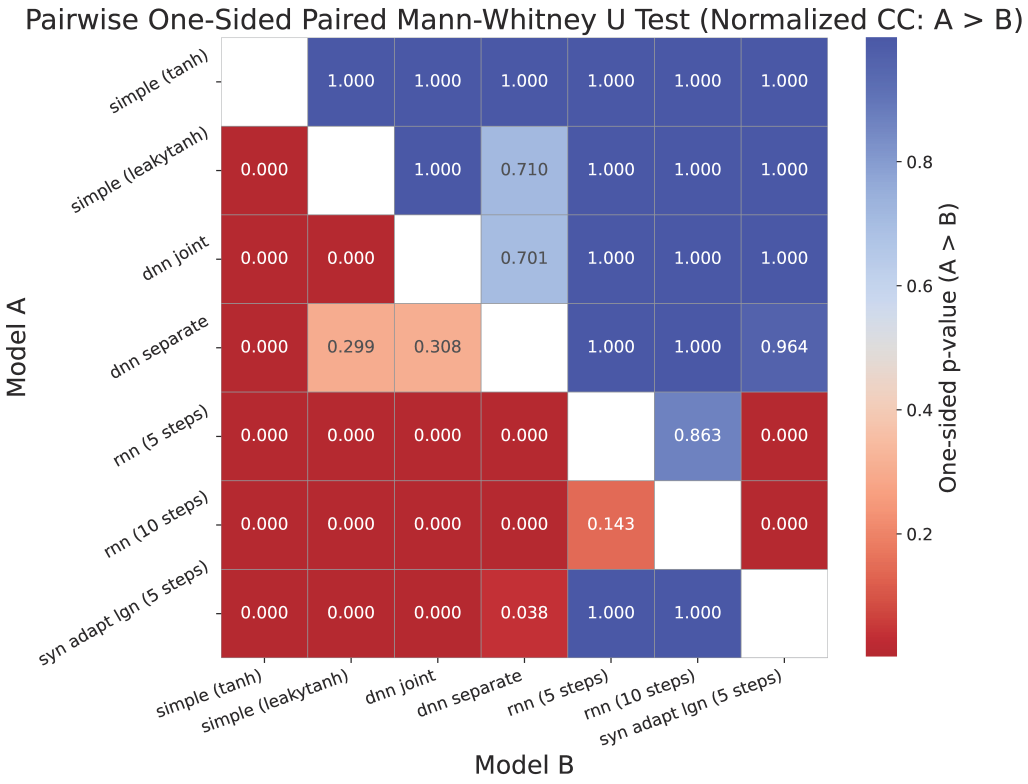

Figure 4.18 Heatmap of p-values from one-sided pairwise Mann-Whitney U tests on normalized cross-correlation values across model variants. Each tile represents a test of whether the row model performs significantly better than the column model. Values below the significance threshold ($\alpha = 0.05$) reject the null hypothesis that both models perform equally or that the row model performs worse.

components and biologically inspired features enhance model performance.

However, two models deviate from this trend. The dnn separate model, which has already been noted for its high performance variability, shows inconsistent statistical superiority. Similarly, the syn adapt lgn (5 steps) model underperforms in comparison to RNN neuronal modules models that lack synaptic depression. As previously discussed, this may be attributed to limitations in hyperparameter tuning, dataset diversity, and computational resources.

In summary, the Mann-Whitney U test provides statistical confirmation that, overall, more complex and biologically grounded models tend to perform better. Exceptions to this trend appear to be primarily due to technical and experimental constraints rather than inherent limitations of the model architectures themselves.

4.4 Analysis of Model Limitations and Improvement Opportunities

Up to this point, we have compared various model architectures and concluded that, with the exception of the synaptic depression variant, increasing biological realism generally enhances model performance. Nonetheless, certain shortcomings remain, particularly in capturing the temporal dynamics of spontaneous neuronal activity. In this section, we examine several contributing factors to these performance gaps and explore potential approaches for improvement.

4.4.1 Free vs Teacher-Forced Prediction Analysis

We begin by comparing two evaluation strategies: free prediction, where the model generates sequences autonomously starting from initial conditions without external corrections, and teacher-forced prediction, where the model's hidden state is reset at each time step using the ground truth neuronal responses. In previous evaluations, we relied solely on free prediction, allowing errors in the model's internal state to accumulate over time. By introducing teacher-forced prediction, we aim to assess whether these accumulated internal state errors significantly impact performance. This comparison focuses primarily on the top-performing TBPTT-trained models, which not only achieve superior predictive accuracy but also maintain a higher degree of biological plausibility.

Figure 4.19 presents the distribution of Pearson correlation coefficients between model predictions and target temporal spiking dynamics curves, separated by layer and prediction mode, excluding the simple (tanh) model.

As observed previously, the dnn separate model exhibits inconsistent behavior. Interestingly, performance varies across layer types. TBPTT-trained models consistently outperform others across majority of layers, although their advantage

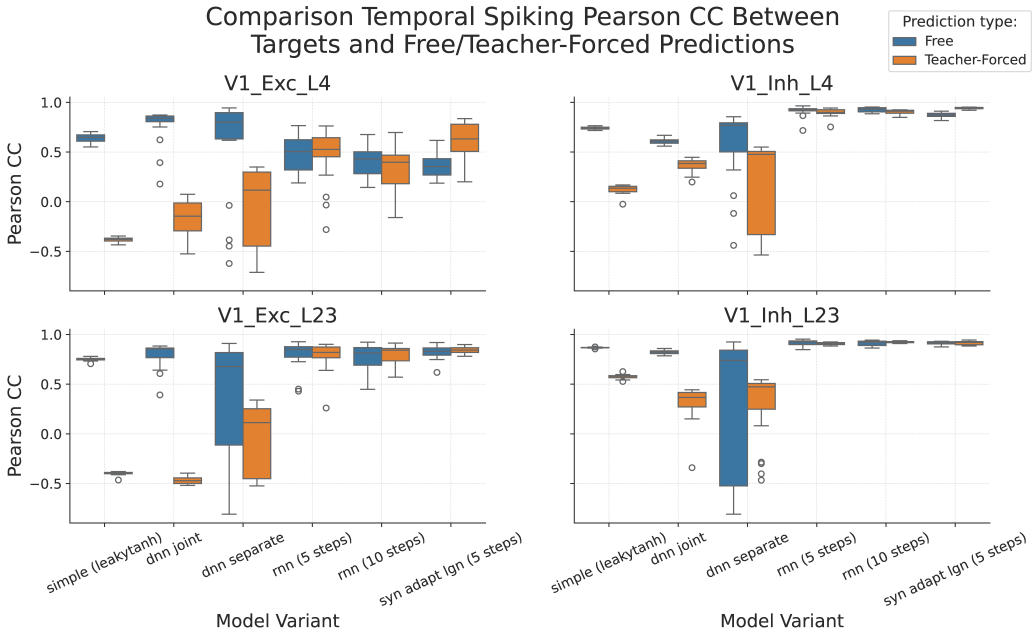

Figure 4.19 Distribution of Pearson correlation coefficients between predicted and target temporal spiking dynamics curves, shown separately for each layer across all model variants.

is less significant in excitatory layers. This discrepancy may be attributed to the relatively sparse activity of excitatory neurons. Since non-TBPTT models tend to approximate mean responses well, they may outperform TBPTT models in these layers. In contrast, TBPTT models achieve superior performance in inhibitory layers, which exhibit higher spiking activity.

Another notable observation is that teacher-forced predictions often match or slightly exceed the performance of free predictions in TBPTT-trained models. This result is both surprising and encouraging: one would expect teacher-forcing to consistently improve predictions by minimizing errors accumulated through imperfect hidden state updates over time. The fact that free and teacher-forced predictions perform similarly suggests that the model has successfully learned to capture the temporal dynamics of the system, maintaining accurate internal representations even without external correction.

Interestingly, the largest improvement under teacher-forced prediction is seen in the synaptic depression model. This suggests that the model may not be effectively learning temporal dependencies, and that extending the TBPTT horizon or increasing training duration could enhance performance.

Conversely, non-TBPTT models perform worse under teacher-forced evaluation. While one might expect hidden state resets to improve performance, given

their similarity to the training procedure, we hypothesize that these resets may disrupt the model's predictions during the high-activity early stimulus phase. Because these models primarily learn mean responses, abrupt resets may distort predictions away from expected averages, resulting in decreased the population spike count CC.

Teacher-Forced Temporal Spiking Dynamics Curves on TBPTT Model Variants

In this section, we narrow our focus to the TBPTT-trained model variants. We compare their temporal spiking dynamics curves under both free prediction and teacher-forced prediction modes, alongside the target temporal spiking dynamics curves. This comparison is illustrated in Figure 4.20.

From the temporal spiking dynamics curves, we observe a modest improvement in temporal prediction accuracy during the early stages of natural stimulus presentation and at the transition to the blank stimulus phase. However, the models still struggle to replicate the elevated spontaneous activity observed in the blank phase of the target data. There is, however, a subtle increase in dynamic range during these stages, which may indicate potential for further improvement.

One notable discrepancy is observed in the teacher-forced predictions of RNN variants without synaptic depression, particularly in layer IV, where predictions deviate from the target during the later stages of natural stimulus presentation.

An especially interesting case is the teacher-forced output of the synaptic depression model. This variant demonstrates the strongest overall alignment with the target temporal spiking dynamics curve across all TBPTT-trained models—a result also reflected in the population spike count correlation coefficients shown in Figure 4.19. This performance may indicate that, despite the training challenges discussed previously, the model is capable of superior temporal modeling when appropriately guided.

Drift Between Temporal Spiking Dynamics Curves of Teacher-Forced and Free Predictions

In the final part of this section, we examine the difference between teacher-forced and free prediction temporal spiking dynamics curves, referred to here as *drift*. This metric quantifies how model behavior diverges over time when guided by ground truth states versus when operating autonomously. The drift curves for each layer are shown in Figure 4.21.

The greatest drift is observed during the initial natural stimulus phase, the period characterized by the highest neuronal activity. This pattern holds across all tested TBPTT models and highlights the challenge of modeling highly dynamic

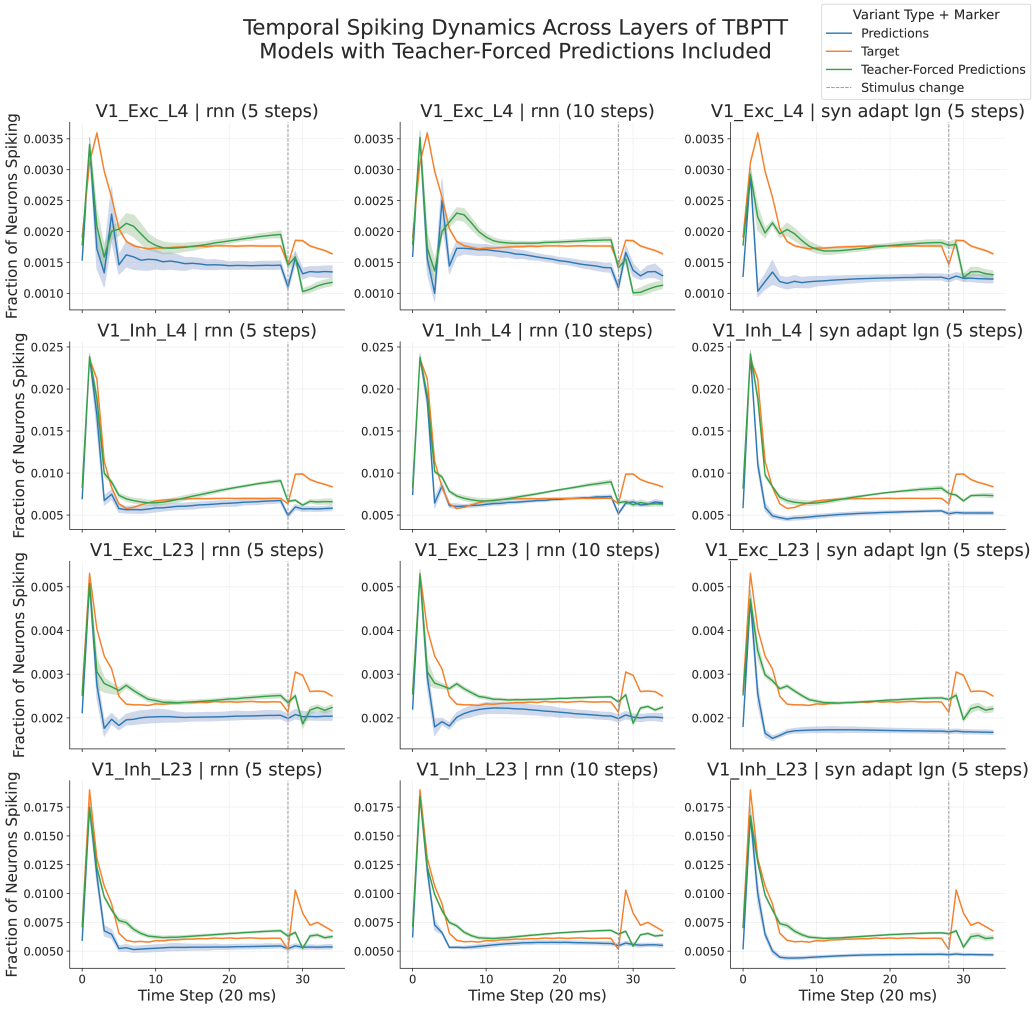

Figure 4.20 Mean temporal spiking dynamics curves of all TBPTT-trained model variants under both free and teacher-forced prediction conditions, compared to the target temporal spiking dynamics curve. Curves are averaged across all trials and model subsets. Error bars represent standard deviation. The dashed vertical line indicates the transition from natural to blank stimuli.

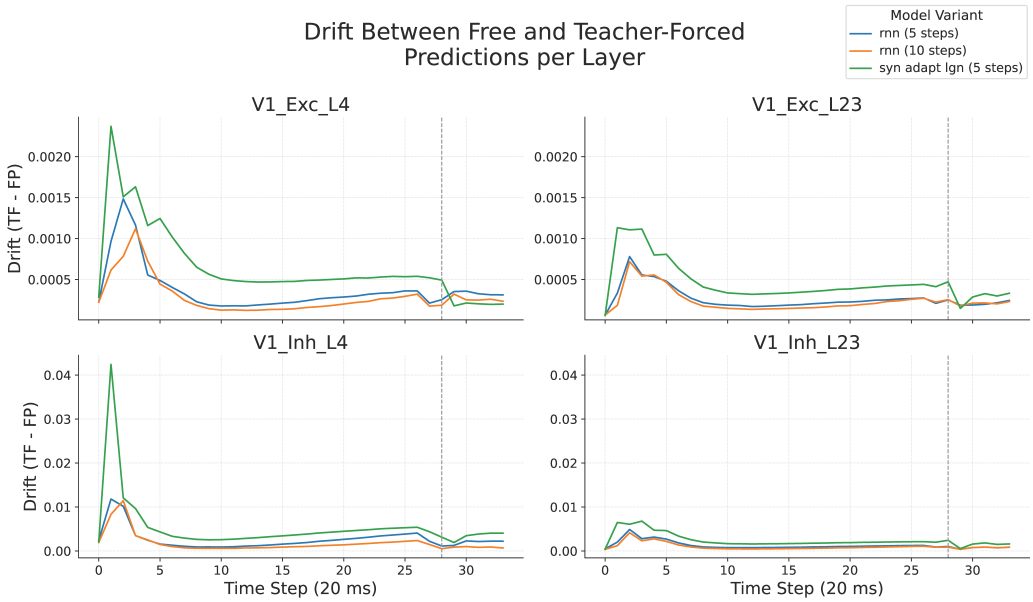

Figure 4.21 Temporal evolution of the difference ("drift") between teacher-forced and free prediction temporal spiking dynamics curves across the experimental timeline, shown separately for each model layer. Results are averaged across all trials for TBPTT-trained models.

neuronal responses. Notably, the synaptic depression model exhibits the most pronounced drift throughout the experiment, reinforcing our earlier hypothesis regarding suboptimal training in this variant.

However, a promising finding is that the drift in all tested models stabilizes in the later stages of the stimulus sequence. This indicates that the models can adequately customize their responses and capture more stable temporal patterns once the initial burst of activity subsides. The discrepancy between teacher-forced and free predictions appears to be most critical in the highly dynamic early phases.

These observations suggest that the synaptic depression model holds strong potential for modeling biologically realistic temporal dynamics. Optimizing its training procedure, potentially through extended training, better hyperparameter tuning, or alternative optimization strategies, may substantially improve its performance.

4.4.2 Hyperparameter Grid Search

This section provides a brief overview of the hyperparameter grid search procedure conducted to support model evaluation. Due to computational constraints, all grid searches were performed on a single model subset variant different from the

lr	N-CC	P-CC
0.000008	0.350904	0.285114
0.000005	0.350027	0.284416
0.000010	0.270970	0.220144
0.000025	0.187077	0.151972
0.000050	0.114566	0.093056
0.000500	0.097858	0.079533
0.000075	0.077174	0.062676
0.000100	0.037527	0.030460

Table 4.8 Grid Search Summary for simple (tanh) Model. Summary of normalized (N-CC) and Pearson (P-CC) correlation results for different learning rates (lr).

lr	N-CC	P-CC
0.000100	0.878693	0.714005
0.000075	0.877059	0.712670
0.000050	0.872625	0.709055
0.000500	0.794590	0.645656
0.000025	0.294032	0.238913
0.000010	0.010441	0.008462
0.000005	0.001354	0.001069
0.000008	0.000050	0.000018

Table 4.9 Grid Search Summary for simple (leakytanh) Model. Summary of normalized (N-CC) and Pearson (P-CC) correlation results for different learning rates (lr).

subsets selected for evaluation. Details regarding the selection of the validation dataset are outlined in Section 3.2.4.

Simple Model Grid Search

Given the lightweight nature of the simple model in terms of both architecture and computational requirements, we were able to perform a comprehensive grid search over a range of learning rates. The training for each configuration was conducted for 10 epochs. Results for the simple (tanh) and simple (leakytanh) activation variants are presented in Tables 4.8 and 4.9, respectively.

Although no particularly surprising patterns emerged during the grid search, one notable outcome was that the optimal learning rate for the simple (leaky-tanh)variant was significantly higher than that for the tanh variant. Interestingly, the performance trends with respect to learning rate were nearly opposite: the tanh model performed best with lower learning rates, while the leakytanh variant achieved better performance with higher ones.

lr	n-ls	n-nl	n-res	N-CC	P-CC
0.000010	10	3	True	0.880114	0.715163
0.000010	10	7	True	0.879289	0.714492
0.000008	10	3	True	0.878549	0.713885
0.000008	10	7	True	0.878224	0.713618
0.000008	10	5	True	0.877150	0.712746
0.000010	5	5	True	0.876449	0.712174
0.000010	5	7	True	0.871925	0.708491
0.000010	10	5	True	0.871830	0.708410
0.000008	5	5	True	0.866316	0.703933
0.000010	5	3	True	0.860474	0.699177
0.000008	5	7	True	0.857937	0.697110
0.000010	5	5	False	0.854195	0.694092
0.000010	10	3	False	0.825813	0.670711
0.000008	10	3	False	0.813353	0.660599
0.000010	10	5	False	0.798817	0.648800
0.000010	5	3	False	0.766755	0.623028
0.000008	5	7	False	0.759580	0.617220
0.000008	10	7	False	0.751363	0.610289
0.000008	5	5	False	0.723905	0.587979
0.000010	10	7	False	0.722965	0.587225
0.000008	10	5	False	0.626814	0.509143
0.000008	5	3	True	0.295512	0.240123
0.000008	5	3	False	0.008394	0.006809
0.000010	5	7	False	-0.259292	-0.210666

Table 4.10 Grid Search Summary for dnn joint Model. Summary of normalized (N-CC) and Pearson (P-CC) correlation results for different learning rates (lr), neuron module layer size (n-ls), number of layers of neuron module (n-nl) and presence of residual connection in neuron module (n-res).

Ultimately, we selected a learning rate of 0.0000075 for the tanh variant and 0.000075 for the leakytanh variant in our evaluation experiments. These choices were based on a combination of grid search results and empirical insights gained during model development, and differ only slightly from the highest-performing configurations.

DNN Joint Model Grid Search

We extended our hyperparameter grid search to the dnn joint model, using a broader set of parameters as outlined in Table 4.10. Each configuration was trained for 10 epochs using the same model subset variant as in grid search for the simple model.

The grid search results clearly demonstrate the benefit of residual connections

within the neuron modules. Residual-enabled models consistently outperformed their non-residual counterparts in both normalized and Pearson correlation metrics. Empirically, we also observed enhanced training stability, reduced overfitting, and more consistent convergence behavior. Based on these findings, we applied residual connections to all modules in subsequent model configurations.

Regarding the number of layers, the results indicate minimal variation in performance across configurations, though five layers showed the most stable performance overall. This matches observations made during earlier development stages, leading us to select five layers for final evaluation.

For layer size, a size of 10 consistently performed better across configurations. While larger layer sizes could potentially offer additional capacity, our empirical testing did not show improvements. Moreover, larger models significantly increase memory requirements, particularly under TBPTT training. Although this is less of a concern for feed-forward DNNs, we opted to retain the layer size of 10 for both performance and efficiency.

We selected a learning rate of 0.00001 for final evaluation based on its consistently strong performance during the grid search. This choice aligns well with our prior empirical observations regarding training dynamics.

Finally, it is important to note that we did not conduct a grid search for the dnn separate model due to computational resource constraints. This omission may partially explain the inconsistent behavior observed in that model during evaluation.

RNN Separate Model Grid Search

We conducted a grid search for the rnn separate model, focusing primarily on the number of TBPTT (Truncated Backpropagation Through Time) time steps. The layer size was fixed at 10 and the number of layers at 3, based on their reliable performance in the dnn joint model grid search. The choice of a lightweight configuration, particularly with only 3 layers, was driven by the high computational and memory demands associated with recurrent modules, especially when using longer TBPTT sequences.

Due to these resource constraints, we limited the search space and acknowledge that the chosen configurations may not represent a global optimum. Each model variant was trained for 40 epochs to better accommodate the slower convergence of RNNs. Results of this grid search are summarized in Table 4.11. Note that we excluded the rnn joint variant from this study, as our goal is to prioritize biologically plausible architectures, and the joint variant conflicts with this aim.

The grid search yielded an unexpected finding: models with higher TBPTT step counts generally performed worse than those with 5 or 10 steps. We hypothesize that this is due to insufficient training time, longer TBPTT sequences

lr	n-tbptt	N-CC	P-CC
0.000030	10	0.933978	0.758967
0.000030	5	0.932902	0.758091
0.000050	10	0.932787	0.757994
0.000010	5	0.914351	0.743027
0.000030	20	0.913769	0.742504
0.000010	10	0.911943	0.741078
0.000050	40	0.911054	0.740291
0.000050	5	0.885195	0.719318
0.000010	20	0.873075	0.709455
0.000050	20	0.829914	0.674317
0.000030	40	0.689550	0.560315
0.000010	40	0.281750	0.228879

Table 4.11 Grid Search Summary for rnn separate Model. Summary of normalized (N-CC) and Pearson (P-CC) correlation results for different learning rates (lr) and number of TBPTT time steps (n-tbptt).

likely require additional epochs to converge. This theory is supported by the performance of the model trained with 40 TBPTT steps and the highest learning rate (0.00005), which still achieved relatively competitive correlation scores.

Nevertheless, given the significant increase in computational cost associated with longer TBPTT sequences, we selected 5 and 10 steps as our evaluation standards. These configurations offered the best balance of performance and feasibility. As for the learning rate, 0.00003 was chosen for final evaluation, based on its consistent top performance in the tested scenarios.

Synaptic Depression Grid Search

In the final section dedicated to grid search, we focus on the hyperparameters for the synaptic depression module. This part of the analysis was significantly limited by computational resources. Performing a comprehensive grid search on models with full synaptic depression, applied across all connected pairs of neuronal populations, is computationally intensive, particularly when applied with longer TBPTT sequences.

Although our original goal was to train each configuration for 40 epochs, we were only able to train for 20 epochs due to time and memory constraints. Similarly, the number of TBPTT steps had to be limited to 5 and 10. For the RNN neuron module, we used 3 layers of size 10 (as in the RNN grid search). To further manage resource demands, the synaptic depression modules was configured with just 2 layers of size 10. Grid search results are presented in Table 4.12.

The results reveal that the full synaptic depression model achieved a normal-

lr	n-tbptt	N-CC	P-CC
0.000030	5	0.904516	0.734752
0.000050	10	0.858861	0.697665
0.000010	5	0.826371	0.671237
0.000010	10	0.728944	0.592099
0.000050	5	0.678222	0.550937
0.000030	10	-0.016808	-0.013631

Table 4.12 Grid Search Summary for synaptic depression Model. Summary of normalized (N-CC) and Pearson (P-CC) correlation results for different learning rates (lr) and number of TBPTT time steps (n-tbptt).

ized CC of 0.90 using only 5 TBPTT steps and 20 training epochs. This is an encouraging outcome, suggesting that with longer training durations and more extensive hyperparameter tuning, the model could offer significant performance improvements.

This incomplete grid search also partially explains the poor performance of the syn adapt lgn (5 steps) model. In that case, synaptic depression was applied only to LGN connections, a simplification necessitated by resource constraints. During evaluation, the LGN-only variant demonstrated slower learning and required more epochs to reach competitive correlation values.

Moreover, the limited training duration and use of only 5 TBPTT steps likely contributed to underfitting. This is further supported by the observation that teacher-forced predictions were significantly better for the synaptic depression model, a trend not observed in RNN variants without synaptic depression. These findings collectively suggest that synaptic depression holds substantial promise, and that future work with better computational resources could unlock its full potential.

4.4.3 Impact of Training Dataset Size on Model Performance

In the final part of our experimental analysis, we briefly examine how the size of the training dataset influences model performance, as measured by normalized cross-correlation (CC). For this investigation, we utilized the dnn joint model due to its relatively low computational cost and its inclusion of a neuron module in the architecture. Each model was trained for 10 epochs using the same setup as described in Table 4.6, and trained on the same model subset as used in the grid search.

Our initial goal was to evaluate the model's performance across the full range of training dataset sizes, defined comprehensively in Section 3.2.4, increasing in 2% increments. However, due to computational limitations, we were only able to

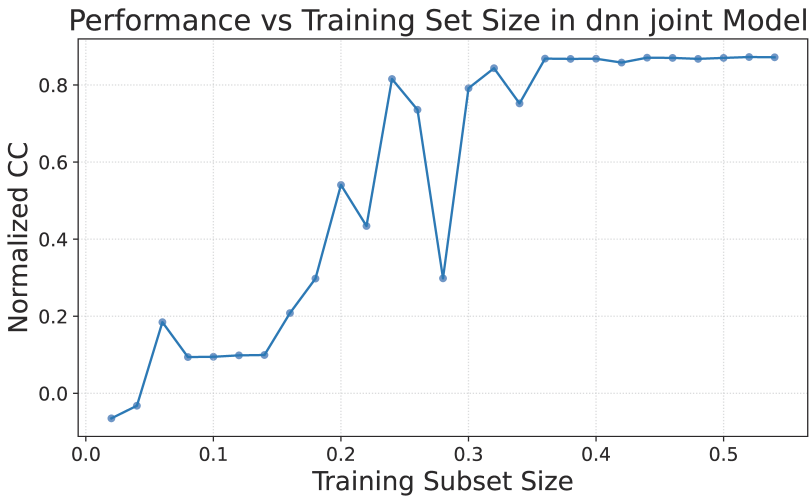

Figure 4.22 Dependency of normalized CC on training dataset size for the dnn joint model using the hyperparameter configuration from Table 4.6. Due to computational constraints, only dataset sizes up to 54% of the original training data were tested.

evaluate model performance for training dataset ratios in the range $[0.02, 0.54]$ with step increments of 0.02. The results are visualized in Figure 4.22.

The results indicate that normalized CC values stabilize around 36% of the training dataset. Beyond this point, additional data does not appear to yield significant improvements in performance. This finding suggests that increasing the size of the training dataset beyond this threshold is unlikely to provide meaningful benefits for the dnn joint model, at least within the constraints of a feed-forward architecture.

That said, this analysis was limited to a single model type. Further investigation is needed to determine whether this trend holds across more complex architectures, particularly those incorporating recurrent or adaptive components. Additionally, our dataset is derived from a limited range of experimental conditions. Exploring alternative stimulus types, such as natural movies or other paradigms discussed in Section 2.1.2, may offer further opportunities to enhance model generalization and performance.

Conclusion

This thesis focused on developing a recurrent neural network (RNN) that mimics the spiking activity of a spiking neural network (SNN) modeling the cat's primary visual cortex (V1), as proposed by Antolík et al. [10], while incorporating known biological constraints of the system. The overarching goal was to construct an explainable model of V1 using deep neural network techniques. Throughout the thesis, we arrived at several key findings.

We first implemented a base RNN model whose architecture directly maps each neuron in the SNN to a corresponding neuron in the RNN, reflecting a simplified version of the biological V1's layered structure. Due to computational constraints, we limited the model to randomly selected subsets of neurons from each population and reduced the temporal resolution from 1 ms to 20 ms. We further introduced several biologically inspired modules to improve both model interpretability and performance. These included shared neural network modules per neuron, acting as trainable, complex replacements for standard activation functions, and synaptic depression modules that modify neuronal inputs in a manner analogous to biological synaptic depression.

Our results demonstrate that the addition of biologically motivated modules improves model performance both in terms of general predictive metrics (such as normalized cross-correlation) and in capturing the temporal dynamics of neuronal responses, as reflected by temporal behavior curves across neuronal populations. The best-performing model variants showed promising fidelity in reproducing the temporal behavior of the SNN, supporting the notion that integrating anatomical constraints into deep neural network architectures holds significant promise for modeling neural activity.

This hybrid modeling approach, which draws from the strengths of both deep learning and biologically grounded methods, may provide a scalable and interpretable framework for neural system modeling. Deep neural networks offer strong performance across various tasks and benefit from continuous improvements in scalability and training efficiency. By embedding biological constraints into their structure, we gain insights into neural function and mitigate the common criticism of DNNs as "black box" models.

Importantly, this research contributes to the growing field of biologically inspired artificial intelligence. It demonstrates a practical method for combining the computational power of deep learning with the structural interpretability of biological modeling, laying the groundwork for future interdisciplinary applications in computational neuroscience, machine learning, and brain-computer interface development.

However, several limitations emerged. Most notably, the models struggled to accurately predict spontaneous neuronal activity during blank stimulus phases, pointing to incomplete modeling of temporal behavior. In particular, the most biologically complex variant, the model incorporating synaptic depression, performed worse than expected. We attribute this to limited computational resources and the high complexity of the model. Furthermore, although this study benefited from full knowledge of the target model (the SNN), real-world scenarios often lack such ground truth data. Despite this advantage, reproducing the SNN's responses remains a challenging task, underscoring the inherent complexity of modeling the primary visual cortex.

Future Work

There is substantial room for improvement in hyperparameter tuning, particularly for the synaptic depression model. More fine-grained hyperparameter selection, extended training durations, and access to enhanced computational resources may resolve the shortcomings observed in our current implementation.

Once these optimizations are in place, the model should be tested on real biological datasets. This would involve withholding some neuronal responses during training and evaluating the model's ability to predict these unseen targets—closely mimicking conditions typical in neuroscience, where only limited neuronal recordings are available.

Upon successful performance under these conditions, the model should be fine-tuned using biological recordings A strong performance on real biological data could mark a significant advancement in our capacity to model neural systems and improve our understanding of the visual cortex.

Ultimately, we envision integrating this framework with the comprehensive SNN model of V1 by Antolík et al. [10], thus enabling deeper and more interpretable analyses of the visual system.

Conclusively, this work represents a foundational step toward merging computational power with biological fidelity, bringing us closer to truly interpretable models of brain function.

Bibliography

[1] Thomas Trappenberg. *Fundamentals of Computational Neuroscience*. English. Google Books ID: n53FAgAAQBAJ. OUP Oxford, 2009, p. 417. ISBN: 978-0-19-102944-8.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: 25 (2012). Ed. by F. Pereira et al. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[3] Qing Li et al. "Medical Image Classification with Convolutional Neural Network". In: *2014 13th International Conference on Control Automation Robotics and Vision (ICARCV)*. 2014, pp. 844–848. DOI: 10.1109/ICARCV.2014.7064414.

[4] Alessia Celeghin et al. "Convolutional Neural Networks for Vision Neuroscience: Significance, Developments, and Outstanding Issues". In: *Frontiers in Computational Neuroscience* 17 (2023). Publisher: Frontiers. ISSN: 1662-5188. DOI: 10.3389/fncom.2023.1153572. URL: https://www.frontiersin.org/articles/10.3389/fncom.2023.1153572/full.

[5] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "SPIKING NEURAL NETWORKS". In: *International Journal of Neural Systems* 19.04 (2009). PMID: 19731402, pp. 295–308. DOI: 10.1142/S0129065709002002. eprint: https://doi.org/10.1142/S0129065709002002. URL: https://doi.org/10.1142/S0129065709002002.

[6] Kashu Yamazaki et al. "Spiking Neural Networks and Their Applications: A Review". In: *Brain Sciences* 12.7 (2022). ISSN: 2076-3425. DOI: 10.3390/brainsci12070863. URL: https://www.mdpi.com/2076-3425/12/7/863.

[7] E.M. Izhikevich. "Which model to use for cortical spiking neurons?" In: *IEEE Transactions on Neural Networks* 15.5 (2004), pp. 1063–1070. DOI: 10.1109/TNN.2004.832719.

[8] Risto Miikkulainen et al. *Computational Maps in the Visual Cortex*. English. Google Books ID: xk399ViKHXEC. Springer Science & Business Media, 2006, p. 547. ISBN: 978-0-387-28806-2.

[9] Cristopher M. Niell and Massimo Scanziani. "How Cortical Circuits Implement Cortical Computations: Mouse Visual Cortex as a Model". In: *Annual Review of Neuroscience* 44.Volume 44, 2021 (2021), pp. 517–546. ISSN: 1545-4126. DOI: https://doi.org/10.1146/annurev-neuro-102320-085825. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-neuro-102320-085825.

[10] Ján Antolík et al. "A comprehensive data-driven model of cat primary visual cortex". In: *PLOS Computational Biology* 20.8 (Aug. 2024), pp. 1–42. DOI: 10.1371/journal.pcbi.1012342. URL: https://doi.org/10.1371/journal.pcbi.1012342.

[11] David H. Hubel and Torsten N. Wiesel. "RECEPTIVE FIELDS AND FUNCTIONAL ARCHITECTURE IN TWO NONSTRIATE VISUAL AREAS (18 AND 19) OF THE CAT". In: *Journal of Neurophysiology* 28.2 (1965). PMID: 14283058, pp. 229–289. DOI: 10.1152/jn.1965.28.2.229. eprint: https://doi.org/10.1152/jn.1965.28.2.229. URL: https://doi.org/10.1152/jn.1965.28.2.229.

[12] Mark Bear, Barry Connors, and Michael A. Paradiso. *Neuroscience: Exploring the Brain, Enhanced Edition*. English. Jones & Bartlett Learning, 2020, p. 1018. ISBN: 978-1-284-21128-3.

[13] Rainer Goebel, LARS Muckli, and Dae-Shik Kim. "Visual system". In: *The Human Nervous System Elsevier, San Diego* (2004), pp. 1280–1305.

[14] L. F. Abbott et al. "Synaptic Depression and Cortical Gain Control". In: *Science* 275.5297 (1997), pp. 221–224. DOI: 10.1126/science.275.5297.221. eprint: https://www.science.org/doi/pdf/10.1126/science.275.5297.221. URL: https://www.science.org/doi/abs/10.1126/science.275.5297.221.

[15] Chang Yaramothu and Tara L. Alvarez. "Short-term modification of vergence ramp eye movements in the convergent direction". In: *2014 40th Annual Northeast Bioengineering Conference (NEBEC)*. 2014, pp. 1–2. DOI: 10.1109/NEBEC.2014.6972984.

[16] D. J. Felleman and D. C. Van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex". In: *Cerebral Cortex* 1.1 (1991), pp. 1–47. ISSN: 1047-3211. DOI: 10.1093/cercor/1.1.1-a.

[17] Richard S. Snell and Michael A. Lemp. *Clinical Anatomy of the Eye*. English. Google Books ID: Z8DRCwAAQBAJ. John Wiley & Sons, 2013, p. 434. ISBN: 978-1-118-69101-4.

[18] James H Schwartz et al. *Principles of neural science*. Elsevier New York, 1991. ISBN: 978-0-07-112000-5.

[19] Dale Purves et al. *Neurosciences*. French. Google Books ID: JDSZDwAAQBAJ. De Boeck Supérieur, 2019, p. 964. ISBN: 978-2-8073-1492-4.

[20] Daniel J. Felleman and David C. Van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex". In: *Cerebral Cortex* 1.1 (Jan. 1991), pp. 1–47. ISSN: 1047-3211. DOI: 10.1093/cercor/1.1.1-a. eprint: https://academic.oup.com/cercor/article-pdf/1/1/1/1139605/1-1-1.pdf. URL: https://doi.org/10.1093/cercor/1.1.1-a.

[21] Tim C Kietzmann, Patrick McClure, and Nikolaus Kriegeskorte. "Deep Neural Networks in Computational Neuroscience". In: *bioRxiv* (2018). DOI: 10.1101/133504. eprint: https://www.biorxiv.org/content/early/2018/06/05/133504.full.pdf. URL: https://www.biorxiv.org/content/early/2018/06/05/133504.

[22] Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. English. Google Books ID: fLT4DwAAQBAJ. MIT Press, 2005, p. 477. ISBN: 978-0-262-54185-5.

[23] Daniel A. Butts. "Data-Driven Approaches to Understanding Visual Neuron Activity". In: *Annual Review of Vision Science* 5.Volume 5, 2019 (2019), pp. 451–477. ISSN: 2374-4650. DOI: https://doi.org/10.1146/annurev-vision-091718-014731. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-vision-091718-014731.

[24] Matteo Carandini et al. "Do We Know What the Early Visual System Does?" In: *Journal of Neuroscience* 25.46 (2005), pp. 10577–10597. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.3726-05.2005. eprint: https://www.jneurosci.org/content/25/46/10577.full.pdf. URL: https://www.jneurosci.org/content/25/46/10577.

[25] E J Chichilnisky. "A simple white noise analysis of neuronal light responses". In: *Network: Computation in Neural Systems* 12.2 (May 2001), p. 199. DOI: 10.1088/0954-898X/12/2/306. URL: https://dx.doi.org/10.1088/0954-898X/12/2/306.

[26] Vargha Talebi and Curtis L. Baker. "Natural versus Synthetic Stimuli for Estimating Receptive Field Models: A Comparison of Predictive Robustness". In: *Journal of Neuroscience* 32.5 (2012), pp. 1560–1576. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.4661-12.2012. eprint: https://www.jneurosci.org/content/32/5/1560.full.pdf. URL: https://www.jneurosci.org/content/32/5/1560.

[27] Saurabh Sonkusare, Michael Breakspear, and Christine Guo. "Naturalistic Stimuli in Neuroscience: Critically Acclaimed". In: *Trends in Cognitive Sciences* 23.8 (2019). Publisher: Elsevier, pp. 699–714. ISSN: 1364-6613, 1879-307X. DOI: 10.1016/j.tics.2019.05.004. URL: https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(19)30127-5.

[28] Konstantin-Klemens Lurz et al. "Generalization in data-driven models of primary visual cortex". In: *bioRxiv* (2021). DOI: 10.1101/2020.10.05.326256. eprint: https://www.biorxiv.org/content/early/2021/04/06/2020.10.05.326256.full.pdf. URL: https://www.biorxiv.org/content/early/2021/04/06/2020.10.05.326256.

[29] Michael C.-K. Wu, Stephen V. David, and Jack L. Gallant. "COMPLETE FUNCTIONAL CHARACTERIZATION OF SENSORY NEURONS BY SYSTEM IDENTIFICATION". In: *Annual Review of Neuroscience* 29.Volume 29, 2006 (2006), pp. 477–505. ISSN: 1545-4126. DOI: https://doi.org/10.1146/annurev.neuro.29.051605.113024. URL: https://www.annualreviews.org/content/journals/10.1146/annurev.neuro.29.051605.113024.

[30] Ethem Alpaydin. *Introduction to Machine Learning, Fourth Edition*. English. Google Books ID: uZnSDwAAQBAJ. MIT Press, 2020, p. 709. ISBN: 978-0-262-35806-4.

[31] J A Movshon, I D Thompson, and D J Tolhurst. "Receptive field organization of complex cells in the cat's striate cortex." In: *The Journal of Physiology* 283.1 (1978), pp. 79–99. DOI: https://doi.org/10.1113/jphysiol.1978.sp012489. eprint: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1978.sp012489. URL: https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1978.sp012489.

[32] Deepankar Mohanty, Benjamin Scholl, and Nicholas J. Priebe. "The accuracy of membrane potential reconstruction based on spiking receptive fields". In: *Journal of Neurophysiology* 107.8 (2012). PMID: 22279194, pp. 2143–2153. DOI: 10.1152/jn.01176.2011. eprint: https://doi.org/10.1152/jn.01176.2011. URL: https://doi.org/10.1152/jn.01176.2011.

[33] Robert Shapley. "Linear and nonlinear systems analysis of the visual system: Why does it seem so linear?: A review dedicated to the memory of Henk Spekreijse". In: *Vision Research* 49.9 (2009). From Retinal and Cortical Circuitry to Clinical Application In memory of Henk Spekreijse - a life dedicated to Vision Research, pp. 907–921. ISSN: 0042-6989. DOI: https://doi.org/10.1016/j.visres.2008.09.026. URL: https://www.sciencedirect.com/science/article/pii/S0042698908004653.

[34] Panayiota Poirazi, Terrence Brannon, and Bartlett W. Mel. "Pyramidal Neuron as Two-Layer Neural Network". In: *Neuron* 37.6 (2003). Publisher: Elsevier, pp. 989–999. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(03)00149-1. URL: https://www.cell.com/neuron/abstract/S0896-6273(03)00149-1.

[35] Stephen A. Baccus and Markus Meister. "Fast and Slow Contrast Adaptation in Retinal Circuitry". In: *Neuron* 36.5 (2002). Publisher: Elsevier, pp. 909–919. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(02)01050-4. URL: https://www.cell.com/neuron/abstract/S0896-6273(02)01050-4.

[36] Niru Maheswaranathan et al. "The dynamic neural code of the retina for natural scenes". In: *bioRxiv* (2019). DOI: 10.1101/340943. eprint: https://www.biorxiv.org/content/early/2019/12/17/340943.full.pdf. URL: https://www.biorxiv.org/content/early/2019/12/17/340943.

[37] Daniel A. Butts et al. "Temporal Precision in the Visual Pathway through the Interplay of Excitation and Stimulus-Driven Suppression". In: *Journal of Neuroscience* 31.31 (2011), pp. 11313–11327. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.0434-11.2011. eprint: https://www.jneurosci.org/content/31/31/11313.full.pdf. URL: https://www.jneurosci.org/content/31/31/11313.

[38] Justin Keat et al. "Predicting Every Spike: A Model for the Responses of Visual Neurons". In: *Neuron* 30.3 (2001). Publisher: Elsevier, pp. 803–817. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(01)00322-1. URL: https://www.cell.com/neuron/abstract/S0896-6273(01)00322-1.

[39] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". English. In: *Mathematics of Control, Signals, and Systems* 2.4 (1989). Publisher: Springer, pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

[40] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T.

URL: https://www.sciencedirect.com/science/article/pii/089360809190009T.

[41] Yoshua Bengio. "Learning Deep Architectures for AI". In: *Foundations and Trends® in Machine Learning* 2.1 (2009), pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006.

[42] Nikolaus Kriegeskorte. "Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing". In: *Annual Review of Vision Science* 1.Volume 1, 2015 (2015), pp. 417–446. ISSN: 2374-4650. DOI: https://doi.org/10.1146/annurev-vision-082114-035447. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-vision-082114-035447.

[43] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 265–283. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[44] Adam Paszke et al. "Automatic Differentiation in PyTorch". English. In: (2017). Preprint, OpenReview. URL: https://openreview.net/forum?id=BJJsrmfCZ.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". English. In: *Nature* 521.7553 (2015). Publisher: Nature Publishing Group, pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://www.nature.com/articles/nature14539.

[46] Santiago A. Cadena et al. "Deep convolutional models improve predictions of macaque V1 responses to natural images". In: *PLOS Computational Biology* 15.4 (Apr. 2019), pp. 1–27. DOI: 10.1371/journal.pcbi.1006897. URL: https://doi.org/10.1371/journal.pcbi.1006897.

[47] William F. Kindel, Elijah D. Christensen, and Joel Zylberberg. *Using deep learning to reveal the neural code for images in primary visual cortex*. 2017. arXiv: 1706.06208 [q-bio.NC]. URL: https://arxiv.org/abs/1706.06208.

[48] Masoumeh Zareh et al. "A Deep Learning Model of Dorsal and Ventral Visual Streams for DVSD". English. In: *Scientific Reports* 14.1 (2024). Publisher: Nature Publishing Group, p. 27464. ISSN: 2045-2322. DOI: 10.1038/s41598-024-78304-7. URL: https://www.nature.com/articles/s41598-024-78304-7.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[50] Yimeng Zhang et al. "Convolutional Neural Network Models of V1 Responses to Complex Patterns". English. In: *Journal of Computational Neuroscience* 46.1 (2019), pp. 33–54. ISSN: 1573-6873. DOI: 10.1007/s10827-018-0687-7. URL: https://doi.org/10.1007/s10827-018-0687-7.

[51] Ján Antolík et al. "Model Constrained by Visual Hierarchy Improves Prediction of Neural Responses to Natural Scenes". In: *PLOS Computational Biology* 12.6 (June 2016), pp. 1–22. DOI: 10.1371/journal.pcbi.1004927. URL: https://doi.org/10.1371/journal.pcbi.1004927.

[52] Hulusi Kafaligonul, Bruno G. Breitmeyer, and Haluk Öğmen. "Feedforward and feedback processes in vision". In: *Frontiers in Psychology* 6 (2015). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2015.00279. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2015.00279.

[53] Tian-De Shou. "The Functional Roles of Feedback Projections in the Visual System". English. In: *Neuroscience Bulletin* 26.5 (2010), pp. 401–410. ISSN: 1673-7067. DOI: 10.1007/s12264-010-0521-3. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5560353/.

[54] Kohitij Kar et al. "Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior". In: *Nature neuroscience* 22.6 (2019), pp. 974–983. URL: https://www.nature.com/articles/s41593-019-0392-5#citeas.

[55] Larry Medsker and Lakhmi C. Jain. "Recurrent Neural Networks: Design and Applications". English. In: (1999). Google Books ID: ME1SAkN0PyMC, p. 414.

[56] Valerio Mante et al. "Context-Dependent Computation by Recurrent Dynamics in Prefrontal Cortex". English. In: *Nature* 503.7474 (2013). Publisher: Nature Publishing Group, pp. 78–84. ISSN: 1476-4687. DOI: 10.1038/nature12742. URL: https://www.nature.com/articles/nature12742.

[57] H. Francis Song, Guangyu R. Yang, and Xiao-Jing Wang. "Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework". In: *PLOS Computational Biology* 12.2 (Feb. 2016), pp. 1–30. DOI: 10.1371/journal.pcbi.1004792. URL: https://doi.org/10.1371/journal.pcbi.1004792.

[58] Nicolas Y. Masse et al. "Circuit Mechanisms for the Maintenance and Manipulation of Information in Working Memory". English. In: *Nature Neuroscience* 22.7 (2019). Publisher: Nature Publishing Group, pp. 1159–1167. ISSN: 1546-1726. DOI: 10.1038/s41593-019-0414-3. URL: https://www.nature.com/articles/s41593-019-0414-3.

[59] Robert Kim, Yinghao Li, and Terrence J. Sejnowski. "Simple framework for constructing functional spiking recurrent neural networks". In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22811–22820. DOI: 10.1073/pnas.1905926116. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1905926116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1905926116.

[60] David Sussillo et al. "A neural network that finds a naturalistic solution for the production of muscle activity". In: *Nature neuroscience* 18.7 (2015), pp. 1025–1033. URL: https://www.nature.com/articles/nn.4042.

[61] Shreya Saxena et al. "Motor cortex activity across movement speeds is predicted by network-level strategies for generating muscle activity". In: *eLife* 11 (May 2022). Ed. by J Andrew Pruszynski and Joshua I Gold, e67620. ISSN: 2050-084X. DOI: 10.7554/eLife.67620. URL: https://doi.org/10.7554/eLife.67620.

[62] Vishwa Goudar et al. "Schema Formation in a Neural Population Subspace Underlies Learning-to-Learn in Flexible Sensorimotor Problem-Solving". English. In: *Nature Neuroscience* 26.5 (2023). Publisher: Nature Publishing Group, pp. 879–890. ISSN: 1546-1726. DOI: 10.1038/s41593-023-01293-9. URL: https://www.nature.com/articles/s41593-023-01293-9.

[63] Wayne W.M. Soo et al. "Recurrent neural network dynamical systems for biological vision". In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 135966–135982. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/f536d5697b79a9b3b3debbb7a552a7da-Paper-Conference.pdf.

[64] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. English. Google Books ID: Rs4oc7HfxIUC. Cambridge University Press, 2002, p. 498. ISBN: 978-0-521-89079-3.

[65] Guo-qiang Bi and Mu-ming Poo. "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type". In: *Journal of Neuroscience* 18.24 (1998), pp. 10464–10472. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.18-24-10464.1998. eprint: https://www.jneurosci.org/content/18/24/10464.full.pdf. URL: https://www.jneurosci.org/content/18/24/10464.

[66] Natalia Caporale and Yang Dan. "Spike Timing-Dependent Plasticity: A Hebbian Learning Rule". In: *Annual Review of Neuroscience* 31.Volume 31, 2008 (2008), pp. 25–46. ISSN: 1545-4126. DOI: https://doi.org/10.1146/annurev.neuro.31.060407.125639. URL: https://www.annualreviews.org/content/journals/10.1146/annurev.neuro.31.060407.125639.

[67] Robert Haslinger, Gordon Pipa, and Emery Brown. "Discrete Time Rescaling Theorem: Determining Goodness of Fit for Discrete Time Statistical Models of Neural Spiking". In: *Neural Computation* 22.10 (2010), pp. 2477–2506. DOI: 10.1162/NECO_a_00015.

[68] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: *Proceedings of the IEEE* 111.9 (2023), pp. 1016–1054. DOI: 10.1109/JPROC.2023.3308088.

[69] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. "Training Deep Spiking Neural Networks Using Backpropagation". In: *Frontiers in Neuroscience* 10 (Nov. 2016). ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00508. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2016.00508/full (visited on 04/01/2025).

[70] A. L. Hodgkin and A. F. Huxley. "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve". English. In: *The Journal of Physiology* 117.4 (1952), pp. 500–544. ISSN: 0022-3751. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/.

[71] Dean A. Pospisil and Wyeth Bair. "The unbiased estimation of the fraction of variance explained by a model". In: *PLOS Computational Biology* 17.8 (Aug. 2021), pp. 1–36. DOI: 10.1371/journal.pcbi.1009212. URL: https://doi.org/10.1371/journal.pcbi.1009212.

[72] Eric Y. Wang et al. "Towards a Foundation Model of the Mouse Visual Cortex". In: *bioRxiv* (2023). DOI: 10.1101/2023.03.21.533548. eprint: https://www.biorxiv.org/content/early/2023/03/24/2023.03.21.533548.full.pdf. URL: https://www.biorxiv.org/content/early/2023/03/24/2023.03.21.533548.

[73] Anne Hsu, Alexander Borst, and Frédéric E Theunissen. "Quantifying variability in neural responses and its application for the validation of model predictions". In: *Network: Computation in Neural Systems* 15.2 (Jan. 2004), pp. 91–109. DOI: 10.1088/0954-898X_15_2_002. URL: https://doi.org/10.1088/0954-898X_15_2_002.

[74] Jon Touryan, Gidon Felsen, and Yang Dan. "Spatial Structure of Complex Cell Receptive Fields Measured with Natural Images". In: *Neuron* 45.5 (Mar. 2005), pp. 781–791. DOI: 10.1016/j.neuron.2005.01.029. URL: https://www.cell.com/neuron/abstract/S0896-6273(05)00061-9.

[75] Patrick Gill et al. "Sound Representation Methods for Spectro-Temporal Receptive Field Estimation". English. In: *Journal of Computational Neuroscience* 21.1 (Aug. 2006), pp. 5–20. DOI: 10.1007/s10827-006-7059-4. URL: https://doi.org/10.1007/s10827-006-7059-4.

[76] Oliver Schoppe et al. "Measuring the Performance of Neural Models". In: *Frontiers in Computational Neuroscience* 10 (Feb. 2016). ISSN: 1662-5188. DOI: 10.3389/fncom.2016.00010. URL: https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2016.00010/full (visited on 03/31/2025).

[77] Nicolas Fourcaud-Trocmé et al. "How Spike Generation Mechanisms Determine the Neuronal Response to Fluctuating Inputs". In: *Journal of Neuroscience* 23.37 (2003), pp. 11628–11640. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.23-37-11628.2003. eprint: https://www.jneurosci.org/content/23/37/11628.full.pdf. URL: https://www.jneurosci.org/content/23/37/11628.

[78] Pierre Baudot et al. "Animation of Natural Scene by Virtual Eye-Movements Evokes High Precision and Low Noise in V1 Neurons". In: *Frontiers in Neural Circuits* 7 (Dec. 2013). ISSN: 1662-5110. DOI: 10.3389/fncir.2013.00206. URL: https://www.frontiersin.org/journals/neural-circuits/articles/10.3389/fncir.2013.00206/full (visited on 04/04/2025).

[79] J. Papaioannou and A. White. "Maintained activity of lateral geniculate nucleus neurons as a function of background luminance". In: *Experimental Neurology* 34.3 (1972), pp. 558–566. ISSN: 0014-4886. DOI: https://doi.org/10.1016/0014-4886(72)90050-7. URL: https://www.sciencedirect.com/science/article/pii/0014488672900507.

[80] Clermont Beaulieu and Marc Colonnier. "Number of neurons in individual laminae of areas 3B, 4y, and 6aalpha of the cat cerebral cortex: A comparison with major visual areas". In: *Journal of Comparative Neurology* 279.2 (1989), pp. 228–234. DOI: https://doi.org/10.1002/cne.902790206.

eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cne.902790206. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cne.902790206.

[81] C. Beaulieu et al. "Quantitative Distribution of GABA-immunopositive and-immunonegative Neurons and Synapses in the Monkey Striate Cortex (Area 17)". In: *Cerebral Cortex* 2.4 (July 1992), pp. 295–309. ISSN: 1047-3211. DOI: 10.1093/cercor/2.4.295. eprint: https://academic.oup.com/cercor/article-pdf/2/4/295/1116765/2-4-295.pdf. URL: https://doi.org/10.1093/cercor/2.4.295.

[82] Henry Markram et al. "Interneurons of the Neocortical Inhibitory System". In: *Nature Reviews Neuroscience* 5.10 (Oct. 2004), pp. 793–807. ISSN: 1471-0048. DOI: 10.1038/nrn1519. URL: https://www.nature.com/articles/nrn1519 (visited on 04/04/2025).

[83] Zhou Wang and Alan C. Bovik. "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures". In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117. DOI: 10.1109/MSP.2008.930649.

[84] Torsten Söderström. *Errors-in-Variables Methods in System Identification*. Google Books ID: L7FUDwAAQBAJ. Springer, Apr. 2018. 495 pp. ISBN: 978-3-319-75001-9.

[85] Juan Terven et al. *Loss Functions and Metrics in Deep Learning*. 2024. arXiv: 2307.02694 [cs.LG]. URL: https://arxiv.org/abs/2307.02694.

[86] Fabian Sinz et al. "Stimulus domain transfer in recurrent models for large scale cortical population prediction on video". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/9d684c589d67031a627ad33d59db65e5-Paper.pdf.

[87] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[88] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark". In: *Neurocomputing* 503 (2022), pp. 92–108. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.06.111. URL: https://www.sciencedirect.com/science/article/pii/S0925231222008426.

[89] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG]. URL: https://arxiv.org/abs/1811.03378.

[90] P.J. Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[91] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[92] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[93] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL]. URL: https://arxiv.org/abs/1406.1078.

[94] Ronald J. Williams and Jing Peng. "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories". In: *Neural Computation* 2.4 (1990), pp. 490–501. DOI: 10.1162/neco.1990.2.4.490.

[95] Wolf Singer. "Neuronal Synchrony: A Versatile Code for the Definition of Relations?" In: *Neuron* 24.1 (1999). Publisher: Elsevier, pp. 49–65. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(00)80821-1. URL: https://www.cell.com/neuron/abstract/S0896-6273(00)80821-1.

[96] Alex M Lamb et al. "Professor Forcing: A New Algorithm for Training Recurrent Networks". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/16026d60ff9b54410b3435b403afd226-Paper.pdf.

[97] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. ISSN: 00034851. URL: http://www.jstor.org/stable/2236101 (visited on 04/23/2025).

Appendix A

Model Implementation and Evaluation Tools

A significant portion of the work for this thesis was dedicated to preprocessing the dataset from the source SNN model, developing the RNN model, and creating an evaluation framework. All source code related to these functionalities is included in the thesis attachment, accompanied by documentation providing comprehensive descriptions of each tool.

Throughout the project, development was managed using a GitHub version control repository, where the most up-to-date version of the thesis source code remains available. The attachment provided with this thesis is a snapshot of our working branch `model_restruct`. Due to file type restrictions and large file size, a small example dataset was removed from the attachment. For access to an example dataset, the latest project version, or to review the development history, please refer to the project's GitHub repository:[1].

It should be noted that a small portion of the codebase was contributed by another student, Richard Kraus, who worked on enhancing the model architecture by applying spatial constraints. This contribution is primarily encapsulated in the source file `nn_model/connection_learning.py`. However, this functionality was not incorporated into the implementation discussed throughout this thesis and remains an optional plugin for potential future model extensions. Additionally, Richard Kraus made several minor contributions to other parts of the codebase, aimed at improving memory and computational efficiency, as well as aiding in the installation process on the MetaCentrum[2] computational cluster. All contributions can be tracked in the GitHub repository.

The dataset utilized throughout this thesis and the evaluation results are

[1] `https://github.com/dbeinhauer/mcs-source`

[2] `https://metavo.metacentrum.cz/`

stored on the Wintermute cluster of the CSNG MFF CUNI group, with storage details further specified in the attached documentation.