Bachelor's Thesis

Timing distributions of sequence elements for dendritic sequence processing

Lisa Golla

Examiner: Dr. Farbod Nosrat Nezami
Advisers: Prof. Dr. Pascal Nieters

University Osnabrück
Faculty of Human Sciences
Department of Cognitive Science

June 12th, 2024

Writing period

07. 03. 2024 – 12. 06. 2024

Examiner

Dr. Farbod Nosrat Nezami

Advisers

Prof. Dr. Pascal Nieters

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Place, Date	Signature

Abstract

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AP	Action Potential
AUC	Area under curve
Ca^{2+}	Calcium ions
CD	Coincidence Detection
DenRAM	Dendritic RRAM
FN	False Negative
FP	False Positive
FPR	False Positive Rate
HTM	Hierachical Temporal Memory
MCC	Matthew Correlation Coefficient
ms	Milliseconds
Na^{+}	Sodium ions
NMDA	N-Methyl-D-Aspartat
ROC	Receiver Operating Characteristic
RRAM	Resistive Random Access Memory
SRNN	Recurrent Spiking Neural Networks
TN	True Negative
TP	True Positive
TPR	True Positive Rate

Nomenclature

α alphabet - a set of symbols sequences are sampled from

μ Mean

σ Standard Deviation

a minimum values

b maximum values

$\boldsymbol{\tau}$ plateau potential duration

1 Introduction

Context 1-2 paragraphs

In the field of computational neuroscience, researchers are actively investigating dendritic processing as a means to replicate the intricate information integration seen in biological brains. Specifically, it is of interest to find a theoretical model capturing the features of the dendrites in order to understand the role of dendrites in neural computation as well as to construct more efficient and intelligent systems. Whereas there are several approaches modeling already known features of dendrites, it is still unknown and heavily discussed what exactly dendrites are able to compute and which specific role they play in the functionality of neurons (Cuntz et al., 2014; London & Häusser, 2005).

A crucial aspect currently debated in this context is the temporal dynamics of sequence elements in dendrites, which govern essential cognitive functions like memory retrieval and pattern recognition (Branco et al., 2010; Gasparini et al., 2004). While recognizing patterns of spiking activity across various timeframes is an essential and remarkable function of the brain, this aspect is not adequately represented in popular point-neuron models such as leaky integrate-and-fire models, which are limited by fixed timescales of passive membrane potential dynamics (Leugering et al., 2023). Therefore, the need emerges, to conceptualize theoretical frameworks modeling temporal dynamics of sequence elements in dendrites resulting in the ability to detect patterns of spiking activity time-invariant. In conclusion, the topic can be narrowed down to dendritic sequence processing.

Motivation 1-2 paragraphs

warum time invariant detection relevant / robust to timing jitters? warum anwendungen nennen (audio input processing, ...)

Überleitung

recurrent neural networks können problem solven aber sind ineffizient, passive dynamics sind zu statisch : wie in dendrites modellieren löst problem effizient und auch suitable for hardware? perspectives of applications (leugering; tripod neuron, ...)

why would it be interesting to gather information about tau parameter? –> biological information? –> to build more robust sequence detection systems (to jitters)

Robustness definition *Robustness* of time-invariant sequence detection refers to the ability of a detection algorithm to accurately identify patterns or sequences within data, even when the timings of that data change over time. In other words, it measures how well the detection algorithm performs in the presence of timing variations in the data. On the one hand, this knowledge is valuable in terms of sequence detection algorithms that could potentially be optimized. On the other hand, this knowledge is also valuable regarding biology to unravel the intricacies of dendritic sequence processing by doing experiments in a theoretical model and drawing inferences to biological dendrites.

length of plateau unknown; term robust, plateau potentials are suitable for task, bisher unklar wie lange plateau und warum

Research question & Hypotheses

The focus of this research is to investigate the optimal duration of plateau potentials within dendritic computing systems ensuring robust time-invariant sequence detection. The focus of this research is to investigate the robustness to timing jitters in the context of sequence detection within dendritic computing systems modeling dendritic plateau potentials. Specifically, I aim to address the question:" *To what extent does the time-invariant sequence detection ability of dendritic trees with memory remain robust to timing variations? Testing different logical trees with significant variations in timing, this study aims to measure the reliability of dendritic tree evaluations. How effectively does the system operate if the appropriate tau length is identified? Additionally, what factors influence to the system's robustness?"*

I hypothesize that dendritic trees with memory can classify sequences robustly to timing jitters and that there exists a correlation between the length of the plateau and robust time-invariant sequence detection, with an optimal range falling between 50-600 milliseconds, as suggested by existing literature (see literature review in Section 2.1.4. Additionally, I postulate that different logical dendritic trees are still able to robustly classify sequence despite timing jitters, indicating the importance of considering dendritic morphology in optimizing sequence detection mechanisms. Through empirical investigation and computational modeling, this study seeks to illustrate the interplay between dendritic properties and sequence processing capabilities.

Procedure

To answer the given research question, I will adhere the following procedure. First of all, I will provide some literature background concerning biological dendrites, dendritic potentials as well as the relevance of branching structures. It is focused how dendritic sequence processing proceeds in dendrites and what we currently know about it. Connected to that, I will also provide background about the theoretical research, e.g. the problem formulation of sequence detection, how it is solved in modelling, as well as detection systems based on plateau potentials and current neuromorphic hardware employing this approach. Throughout, current knowledge gaps are identified.

Furthermore, in section 3 the experimental setup is presented containing design goals, the technical setup, and an explanation how I computationally modeled dendritic trees, the classifier and trials. Moreover, information are provided regarding the experiment itself like the evaluation metrics, parameters used as well as a detailed description of the procedure and expected outcomes. Afterwards, the section 4 presents the results of the experiments and with given figures the data is analyzed and trends are identified. In the discussion section then 5 the results are interpreted especially in light of the research question and hypothesis. Additionally, limitations and perspectives are shown as well as the findings are connected to current literature. Finally, I will conclude with a summary of key insights.

My main contributions are

- ...

2 Background

The current state of research concerning dendritic computing needs to be considered as multi-faceted. First of all, there is neurobiological research about dendrites that needs to be considered. It deals with the current knowledge about how dendrites work and what they are capable of. Secondly, based on the current state of neurobiological research, there is also work that needs to be considered about how to put these neurobiological findings into a theoretical, simplified, computational model.

In the following this toolkit will be illustrated further as well as (passive +active characterized) regarding dendritic sequence processing it will be explained bla bla oder am anfang bei background schreiben?

2.1 Neurobiological research

2.1.1 Dendrites

In neurobiological research, the understanding of dendrites has undergone a transformative shift. Traditionally viewed as passive conduits, dendrites are now recognized as active and dynamic information processors within neurons. Recent studies have shown their remarkable capacity to influence synaptic integration, modulate plasticity, and contribute significantly to the computation of neural signals (Cuntz et al., 2014). The dynamic interplay between active and passive dendritic properties is emerging as a crucial factor in understanding the intricacies of neural information processing. There are several features of passive and active dendrites known, also referred to as the 'dendritic toolkit' (London & Häusser, 2005).

Passive dendrites are acting as delay lines via dendritic filtering since they linearly filter input signals, thereby labeling specific inputs on distinct dendritic regions by the latency of resulting output spikes (London & Häusser, 2005). This property facilitates parallel processing and local computations, where synaptic inputs not only inject current but also locally change membrane conductance, leading to nonlinear interactions among multiple inputs (Cuntz et al., 2014). These interactions, including those between excitatory synapses and shunting inhibition, resemble Boolean logic operations, suggesting that dendrites may perform complex computational functions similar to modern computers (London & Häusser, 2005).

Active dendrites on the other hand fundamentally alter traditional views of neural computation by introducing feedback mechanisms and amplifying synaptic inputs (Cuntz et al., 2014). Traditionally, neuronal information flow was thought to be unidirectional, from dendrites to soma to axon, however, recent research has revealed the presence of excitable ionic currents in dendrites, supporting dendritic action potentials that propagate in reverse, from soma into dendrites (London & Häusser, 2005). This feedback mechanism fundamentally transforms neurons into closed-loop systems, significantly impacting dendritic function and synaptic plasticity (London & Häusser, 2005). Moreover, active dendrites can amplify synaptic inputs through various mechanisms. These include subthreshold boosting, where inward voltage-dependent currents compensate for signal attenuation, and the generation of local dendritic spikes triggered by synaptic coactivation (London & Häusser, 2005). These spikes, facilitated by voltage-gated channels or N-Methyl-D-Aspartat (NMDA) receptors, substantially enhance computational power by overcoming dendritic attenuation and modulating somatic voltage (Antic et al., 2010; London & Häusser, 2005). Furthermore, in certain neurons like layer 5 cortical pyramidal neurons, global dendritic spikes occur due to the extreme distal placement of synapses. These spikes initiate near branch points, driving a massive dendritic depolarization that communicates with the soma, thereby influencing neuronal firing patterns (London & Häusser, 2005).

2.1.2 Dendritic spikes & Plateau potentials

As elaborated by Antic et al., 2010, 1990 to 2009 can be divided into two decades of dendritic spikes in Cortical Pyramidal Neurons with regard to cortical dendritic physiology. Namely, the first decade 1990-1999 was mainly characterized by findings concerning physiological properties of the thick apical dendrite in pyramidal neurons. It was discovered that apical dendrites contain voltage-gated sodium, potassium, and calcium channels, enabling them to actively transmit Action Potential (AP) backwards through the dendritic tree. Furthermore, beyond their role in action potential back-propagation, these active membrane conductances in the apical region also contribute to the generation of regenerative membrane potentials, commonly referred to as *'dendritic spikes'*, upon adequate stimulation. In the apical dendrite, notable spikes consist of local spikes that do not extend to the soma, along with calcium spikes enabling the most distant synaptic inputs on neocortical pyramidal cells to affect the neuron's overall activity. Additionally, calcium spikes offer significant amplification for layer 1 and layer 2 synaptic inputs. (Antic et al., 2010)

The second decade, 2000-2009 focused on thin branches in Basal, Oblique and Tuft regions. One of the most intriguing aspects of the thin dendrite membrane is its capacity to trigger spikes driven by a ligand-gated receptor channel, namely NMDAr, instead of sodium or calcium voltage-gated channels. When two glutamate ions bind to the glutamate-binding sites of an NMDA receptor channel, a voltage sensitivity is induced, resulting in a distinctive behavior in the channel's current flow. This behavior manifests as a region of negative slope conductance in the current-voltage (I-V) relationship due to the relief of magnesium block. Essentially, under conditions of abundant glutamate availability, the I-V relationship of an NMDA receptor current mimics that of a voltage-gated sodium channel. Consequently, during periods of robust glutamatergic release, NMDA receptor channels generate a regenerative *NMDA spike*, similar to the firing of action potentials by sodium channels upon adequate depolarization. Furthermore, it can be distinguished in between a 'pure' NMDA spike and a NMDA spike / plateau potential. Pure NMDA spikes occur in thin dendrites when two major voltage-gated conductances (Sodium ions (Na^+) and Calcium ions (Ca^{2+})) are pharmacologically blocked with drugs (TTX and Cd21), and the concentration of glutamate surpasses a specific threshold. In the absence of drugs, glutamatergic stimulation above threshold levels leads to a combination of NMDA spikes and activation of dendritic voltage-gated Na1 and Ca21 channels. This results in the generation of *a plateau potential*, which constitutes a complex spike involving multiple complementary conductances that activate in a relatively strict temporal sequence. The plateau potential waveform comprises a fast-onset initial spikelet, followed by a plateau phase and closing with an abrupt collapse at the end of the plateau. For a better understanding of what a pure NMDA spike, plateau potential, and calcium potential entail, waveform and initiation sites are depicted in Figure 1, allowing for a comparison with an action potential. (Antic et al., 2010)

To paraphrase, when synchronous synaptic inputs converge onto a single dendritic branch of layer 5 pyramidal neurons, they induce membrane depolarization, initiating a positive feedback loop. This loop involves NMDA receptor current, which further depolarizes the membrane and recruits additional NMDA-mediated current supported by activated dendritic Na+ and Ca2+ channels (Cuntz et al., 2014; London & Häusser, 2005). This all-or-none occurrence is known as a plateau potential, and its spatial spread is confined to a small section of the dendrite through both active and passive mechanisms (Antic et al., 2010; Leugering et al., 2023). Consequently, plateau potentials are sustained membrane depolarizations in neurons, typically lasting longer than typical action potentials, often due to the activation of voltage-gated calcium channels or other non-linear mechanisms. They exert a local effect within the dendritic tree, influencing nearby synaptic inputs. These potentials are contributing to prolonged excitation and impacting neuronal firing patterns and network activity.

During the evaluation of the literature regarding plateau potentials I was faced with an inconsistent naming convention. Namely, in literature there are different terms to describe the phenomenon of the 'plateau potential' as described above. Examples I encountered with were NMDA spike to refer to the phenomenon, NMDA plateau potential, plateau potential, plateau, Dendritic Sustained Depolarization, Prolonged Dendritic Depolarization, and there may be potentially various other namings. I want to clarify that when talking about the term plateau potential that I refer to the phenomenon as described above. Further, I want to highlight that when evaluating the literature I checked carefully for the phenomenon that is referred to and even if in the literature they use a different term and I refer to this source, I will use the term plateau potential throughout this thesis for consistency purposes and to avoid confusion.

2.1.3 Dendritic sequence processing

As already indicated in the introduction, the brain processes information through spiking activity across various timescales, arising from sensory inputs and internal connectivity. Decoding and detecting the temporal order of these spikes is fundamental for complex behavior, whereas its concrete implementation of neural networks is currently unknown (Cuntz et al., 2014; Leugering et al., 2023). However, research exists suggesting that dendrites are able to perform temporal sequence detection (Cuntz et al., 2014). Strictly speaking, the mechanisms underlying plateau potentials in active dendrites, as discussed in section 2.1.2, involve the capability for coincidence detection (Branco et al., 2010; Gasparini et al., 2004; Leugering et al., 2023; London & Häusser, 2005).

Specifically, dendrites act as delay lines, causing synapses activated along them in different directions to elicit varied responses at the soma (Cuntz et al., 2014). When synapses are activated in sequence from dendrite tip to soma with intervals matching propagation speed, synaptic potentials peak simultaneously at the soma, resulting in a large potential. Conversely, activation in the opposite direction fails to align peaks temporally, leading to a plateau response. Additionally, they exhibit high nonlinearity due to multiple voltage-gated conductances and synaptic input being a conductance

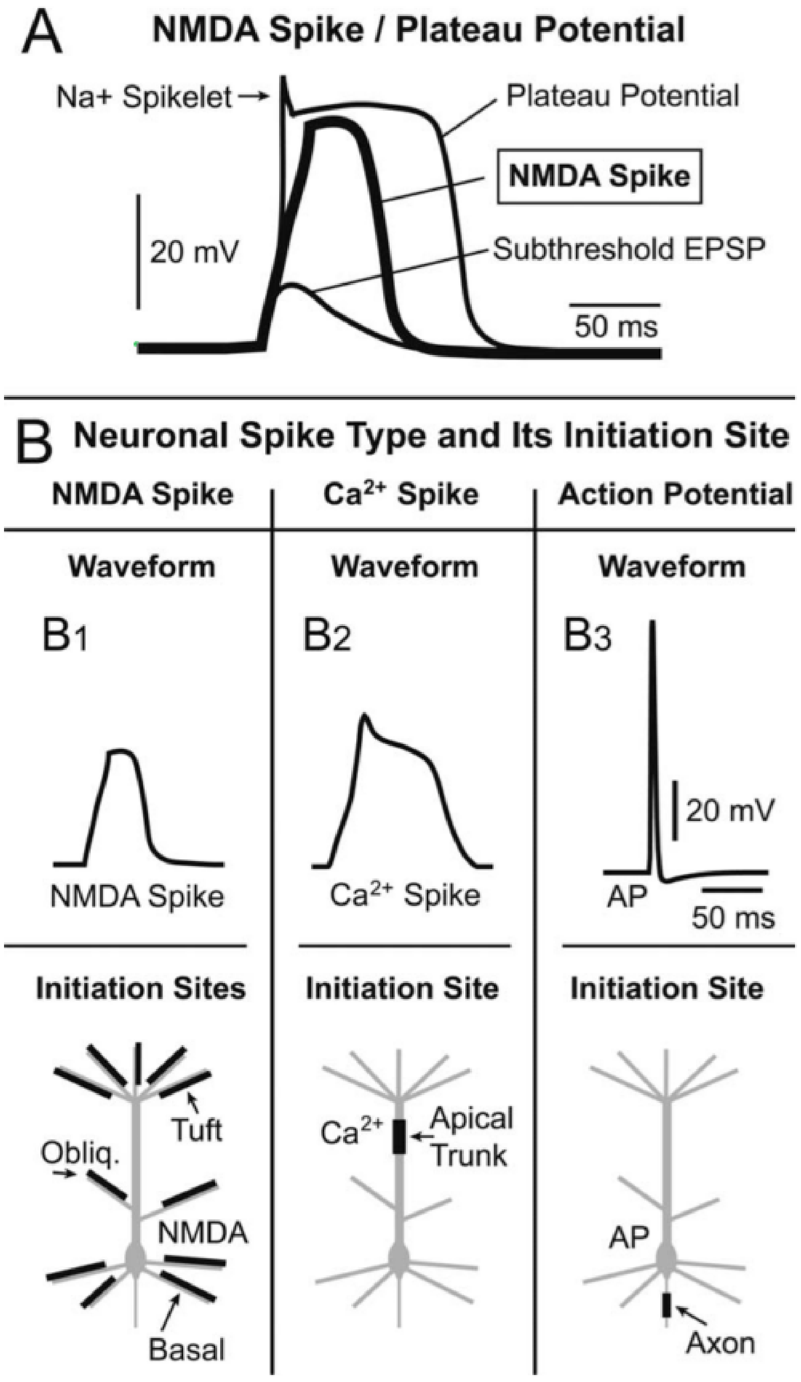


Figure 1: A comparison between dendritic spikes ('pure' NMDA spike, Plateau Potential, Ca2+ spike, and Action Potential) regarding waveform and initiation site. The figure is token from Antic et al., 2010

rather than a pure current source. As argued by Cuntz et. al. (2014) this suggests that dendrites are well-suited for temporal order detection, whereas the mechanism of the plateau potential facilitate this ability.

Moreover, London and Häusser, 2005 specifies this finding by stating that active dendrites hold coincidence detection mechanisms that enable neurons to detect synchronous synaptic inputs as well as these mechanisms operate at both local and global scales within dendritic branches. Through the activation of regenerative inward currents, such as those mediated by NMDA receptors, dendrites can generate local dendritic spikes in response to clustered synaptic inputs. These plateau potentials, amplify coincident synaptic activity and can trigger bursts of action potentials, facilitating rapid information processing. Additionally, dendritic mechanisms allow for the reporting of coincident pre- and postsynaptic activity, further enhancing computational capabilities. These coincidence detection mechanisms are finely tunable, modulated by dendritic geometry and channel properties, and are implicated in various neuronal functions, including synaptic plasticity and cross-layer integration in cortical circuits.

Furthermore, as outlined by Leugering et al., 2023, plateau potentials offer the enduring memory traces essential for sequence processing. When encountering a stimulus, such as coherent synaptic input, the dendrite promptly initiates a plateau potential, preserving a persistent memory trace of the event within the local membrane potential. The crucial mechanism for recognizing sequences of such stimuli on behavioral timescales is the interplay of plateau potentials across adjacent segments, wherein they depolarize neighboring segments. Leugering et al., 2023 further suggests that localized plateau potentials transform the intricate dendritic structures of individual neurons into dependable sequence processors, featuring extensive spatiotemporal receptive fields that remain invariant to timing variations.

In terms of cortical processing, plateau potentials act as the fundamental cellular processes responsible for conducting computations across multiple independent subunits, thereby augmenting the computational capacity and scope of cortical pyramidal cells (Antic et al., 2010). Research suggests that plateau potentials have significant influences on cortical information processing in awake animals, particularly by contributing to functions such as spatiotemporal binding, broadening the dynamic range, and supporting working memory (Antic et al., 2010). Leugering et al., 2023 concludes, that dendritic plateau potentials could potentially complete current models of sequence processing in the brain, whereas the precise role of the plateau potentials in this context is still uncertain.

Concisely, there is evidence suggesting that dendrites are able to perform temporal sequence detection, and in particular it is assumed that so-called plateau potentials might help explain the phenomenon of sequence processing in the brain. Even though research and hypotheses on this exist as presented above, it is crucial to understand that these are founded assumptions and the concrete role as well as the functionality of sequence processing in the brain is still debated.

2.1.4 Plateau potential duration

(im Vergleich "pure" NMDA spike 50-100 ms) full width half amplitude measure skizze literature review präsentieren

First plateau potentials were discovered in cortical pyramidal neurons (Schiller et al., 2000), and from there, extensive research has shown the presence of plateau potentials in hippocampal pyramidal neurons (Suzuki et al., 2008), thalamocortical neurons (Augustinaite et al., 2014), and in layer 5 pyramidal neurons of the frontal cortex (Milojkovic et al., 2004). Whereas originally the findings are based on in vitro experiments, there exists also a range of research showing evidence of NMDA spikes and plateau potentials in vivo (Gao et al., 2021). Despite an increasing research interest, currently there is only little knowledge as well as research on the duration of such a plateau potential. Specifically with regard to dendritic sequence processing, it is unknown whether there is an 'optimal' length or duration correlating with the robustness of time-invariant sequence detection of sensory inputs.

A literature review revealed the current state of research regarding plateau potential duration which can be seen in table 1. I reviewed a range of papers showing evidence or analyzing plateau potentials and searched for information about the duration of the plateaus. In particular, I looked for precise values in Milliseconds (ms), as well as reported factors influencing the duration of the plateau.

explain half-width measure add paper add skizze showing for plateau potential I realized that in the given research field a specific measure is used to describe the plateau duration, namely the half-width measure. As stated in the papers, it quantifies the temporal duration of a signal by determining the time it takes for the signal to decrease to half of its maximum amplitude from its peak. It provides a concise description of the width of a waveform or signal at its halfway point. While the half-width measure offers valuable insights into the temporal dynamics of a signal, it may not fully capture the entire duration of a plateau, as it represents only the time taken for the signal to decrease to half of its maximum amplitude. Therefore, while useful for characterizing waveform width, additional analyses or context may be

Paper	Neuron	Source	Duration (in ms)	Factors +
Milojkovic et al., 2004	cortical pyramidial	Rat	205-499	current strength
Nevian et al., 2007	cortical pyramidal	Rat	-	current strength, glutamate stimulus
Major et al., 2008	cortical pyramidal	Rat	several hundreds	current strength, glutamate stimulus, stronger depolarization
Suzuki et al., 2008	hippocampal pyramidial	Rat	220-610	-
Augustinaite et al., 2014	thalamocortical	Mice	239 - 323	current strength, glutamate stimulus
Gao et al., 2021	cortical pyramidal	Rat	200-500	glutamate stimulation, triggered from distal dendritic segments

Table 1: Durations of plateau potentials in ms stated in given papers. Added are information about the neuron where the plateau potential occured, the source from which the neuron originates, as well as factors reported that increase the plateau potential. A '-' in the duration column indicates, that no value could be found. All duration values are based on half-width measures.

necessary to accurately determine the full duration of plateaus in the given research context. To the best of my knowledge, there is currently only information of the half-width measure values present in research.

report occurence in Gao In particular, the paper by Gao et al., 2021 cited the work by Milojkovic et al., 2004 and Suzuki et al., 2008 and stated that according to their findings, the duration of the plateau lies in between 200-500ms. However, when reading the two papers I realized that the values are based on a half-width measure. This is raising one question for me: How to proceed? When there is a given half-width measure, is it the duration of the plateau then as stated in Gao et al., 2021? Since I am interested in the full duration of a plateau potential, I was wondering, if I need to estimate that value. As far as I understood, the half-width measurement only indicates the point at which the signal decreases to half of its maximum amplitude and may not necessarily mark the end of the plateau and doubling it could offer a rough estimate. Is this correct?

evaluate outcomes of literature review To begin with, my literature review uncovered information regarding the duration of plateau potentials, specifically concerning the half-width measure, which ranged roughly in between 200 and 600 milliseconds. A majority of the papers reported that an increase in current strength and an increase in glutamate stimulation were factors that extended the duration of the plateau. The paper (Gao et al., 2021) singularly reported that the duration of the plateau was longer when triggered from distal dendritic segments. It is striking that in the table there are only papers that employed an in vitro experiment design. I've also analyzed a papers employing in vivo approaches, however there I could not find any information about the half-width measure of the duration of the plateau as well as any other precise measure denoting the length of the plateau. Specifically a challenge was the incoherent naming convention of plateau potentials when analyzing the papers which required in-depth reading.

2.1.5 Branching structures

A classic question in neuroscience revolves around understanding how the structure of dendrites and the different synaptic inputs they receive impact computational processes (London & Häusser, 2005). Traditionally it was thought that dendritic branches only offer a larger surface area for synapses, however current research has shown that the branching structures of dendrites enable far more functional specializations (Ferrante et al., 2013; London & Häusser, 2005; Sinha & Narayanan, 2022; Stuart et al., 2016).

Specifically, in terms of active dendrites, dendritic branches provide the separation and subdivision of afferent inputs, making it possible to process and filter inputs based on where they come from (Sinha & Narayanan, 2022). Additionally, these features help trigger electrical signals in dendrites, recognize when multiple signals arrive at the same time, and enables special communication between dendrites in certain connections (Sinha & Narayanan, 2022). Thus, dendritic arborization is crucial in regulating neural excitability, firing patterns and coincidence detection.(Ferrante et al., 2013; Sinha & Narayanan, 2022).

Subsequently, these findings are particularly relevant in the context of plateau potentials. The localization of plateau potentials within the dendritic tree is attributed to the necessity of external glutamate binding to NMDA receptor channels for their initiation and maintenance (Leugering et al., 2023). More precisely, when a forward propagation event, such as a dendritic spike, encounters the junction where a smaller dendrite merges into a larger one, the inefficiency in propagation becomes apparent due to impedance mismatches (Stuart et al., 2016). This inefficiency at the branch point is critical because it means that local synaptic inputs must interact strategically with the dendritic tree to maintain the spike's propagation. Such local interactions help stabilize the membrane potential, which is crucial for memory encoding, by ensuring that plateau potentials can occur (Stuart et al., 2016). This process underscores how memory mechanisms are intertwined with the structural and functional complexity of the dendritic architecture (Stuart et al., 2016). As a result, the configuration of dendritic arbors emerges as pivotal for plateau computation. Furthermore, more intricate branching patterns can enable individual neurons to detect more complex sequential patterns rendering them a fascinating subject for research (Leugering et al., 2023).

2.2 Theoretical research

2.2.1 Coincidence detection systems

The detection of sequences across various time scales, in literature also referred to as Coincidence Detection (CD) (Bouhadjar et al., 2022; Burger et al., 2023; D'Agostino et al., 2023; Quaresima et al., 2023), has been solved quite differently by theoretical approaches so far. Whereas recurrent populations have necessary memory for sequence detection, they require an excessive number of neurons and need to be finely tuned (D'Agostino et al., 2023). Moreover, they exhibit a high demand for energy. Therefore, research focused on passive dendrite dynamics which are useful for pattern detection, but don't provide proper time invariance as well as it is difficult to build a framework that captures more than 100 ms. The Dendritic RRAM (DenRAM) model (D'Agostino et al., 2023) for example has introduced passive delays in the network helping with reducing the memory footprint and power consumption compared to recurrent populations like Recurrent Spiking Neural Networks (SRNN). Specifically, they achieved processing of 60 ms long sequences in a heartbeat anomaly detection task. However, the approach failed for a keyword spotting task, where delays of up to 500 ms are necessary to reach a desirable accuracy. The time scales are short and rigid making it difficult to adapt to longer dynamic sequences. Additionally, the parameters are fixed and randomly attributed. Work by Hammouamri et al., 2023 however, shows the potential of delay learning in developing accurate and precise models for temporal data processing by training the duration of the delay like a network parameter and maximize it employing deep learning techniques.

Furthermore, delay-based computation is a relatively novel concept and it is likely that the exploitation of the potential of delays has not yet been maximized. Generally, the exploration of the benefits of training delays is a recent trend (D'Agostino et al., 2023; Hammouamri et al., 2023). Nevertheless, as elaborated, neither the excessive number of neurons, finely tuning nor rigid and short timescales are beneficial features for a theoretical approach modeling time invariant and robust sequence detection. Rather, as argued in literature (Bouhadjar et al., 2022; Burger et al., 2023; Leugering et al., 2023; Quaresima et al., 2023), it is more approachable to employ features of active dendrites. In particular, plateau potentials are in discussion to be useful to model robust and time-invariant sequence detection as well as the modeling might help to clarify the role of dendritic action potentials. There are certain approaches modeling the phenomenon of the plateau mainly, especially to gain biological insights (Gao et al., 2021; Milojkovic et al., 2004; Rhodes, 2006; Schiller et al., 2000). Meanwhile these features are also applied to models acting as robust sequence detectors (Bouhadjar et al., 2022; Burger et al., 2023; D'Agostino et al., 2023; Leugering et al., 2023) holding potential for computational use and further expansion in future neuromorphic technologies.

2.2.2 Neuromorphic hardware

Precisely, as highlighted by Cardwell and Chance, 2023, focusing on modeling dendrites will be crucial for leveraging next-generation neuromorphic architectures and applications and requires computational functionalities such as non-linear filtering, direction selectivity, and coincidence detection along with brain-like abilities such as scaling, complexity, computational efficiency, and computational density. Recent progress in neuromorphic computing has achieved passive dendritic compartmentalization in hardware (Kaiser et al., 2022; Yang et al., 2021), as well as non-linear processing in functionally isolated dendrite segments as demonstrated by Intel's Loihi chip (Davies et al., 2018) and the DYNAPSE architecture (Moradi et al., 2018). In addition, the Hierachical Temporal Memory (HTM) algorithm (Hawkins & Ahmad, 2016) has been devised by Billaudelle and Ahmad, 2016 proving that the model can be ported to an analog-digital neuromorphic hardware system. However, the devised model is lacking a solution for online learning, whereas Bouhadjar et al., 2022 is solving this by using local plasticity rules and offers a direct implementation on a neuromorphic hardware system which is still open to be realized. Moreover, using synaptic delays, DenRAM, is the first realization of a spiking neural network with analog dendritic circuits coupled with Resistive Random Access Memory (RRAM) technology (D'Agostino et al., 2023). Based on these findings, it can be said that overall the aim is to build algorithms that are suitable for neuromorphic analogue circuit designs. The focus primarily lies on computational efficiency and complexity as well as to achieve brain-like abilities. Specifically, the realization of a spiking neural network with analog dendritic circuits coupled with active dendritic properties is a still open challenge and offers a promising outlook based on the current state of research (Leugering et al., 2023).

2.2.3 Plateau duration in sequence detection systems

In terms of dendritic sequence processing, an algorithm performing sequence detection should be able to operate in an online fashion to process sequences in real-time and be suitable for real-life applications. While Bouhadjar et al., 2022 proposed an approach solving coincidence detection in an online fashion, the approach can only process fast sequences with inter-stimulus intervals up to roughly 75ms with biological reasonable parameters. This is problematic since behavioral time scales are often larger. The paper supposes an extension of the plateau potential duration which also requires further adaptations in the model which are not solved yet. The paper implemented a plateau potential duration of 50-200 ms, but indicated that this duration can last up to 500 ms. Likewise, the approach by Quaresima et al., 2023 further highlights that the duration of the plateau in their model depends on the dendritic length, strength of synaptic events as well as it can be increased by the number of coincidence inputs and lasts up to 100 ms. The approach by Leugering et al., 2023 modeled the duration of the plateau by a constant of 200 ms. Moreover, Burger et al., 2023 supposes that the longer duration of the plateau potentials can facilitate robustness for time-invariant sequence detection. Consequently, in theoretical model contexts, a plateau potential duration range of 50-500ms can be described. Based on the neurobiological findings regarding the duration of plateaus, it can be inferred that investigating a range between 50 to 600 would be intriguing and whether within this range there exists an 'optimal' duration that enhances the robustness of time-invariant sequence detection.

2.2.4 Branching structures in theoretical frameworks

Currently, with regard to plateau potentials, it is only little known about the exact sub-cellular organization of feature representation and its shaping mechanisms (Moore et al., 2022). In terms of computational models, neurons are often modeled as single compartment models, rather than multi compartment models (Moore et al., 2022). However, dendritic compartments can potentially enrich the capabilities of given models (Moore et al., 2022). In fact, missing experimental constraints are problematic since this implies that there are free parameters which range needs to be explored (Moore et al., 2022).

Nevertheless, there are databases of neural morphologies and algorithms for dendritic remodeling that could, together with biophysically realistic computational modeling, be used to ascertain the impact of active dendritic morphology and to advance computational models in their capacities (Cuntz et al., 2014; Sinha &

Narayanan, 2022). Consequently, the motivation of implementing branching pattern is based on the fact that intricate branching patterns allow individual neurons to detect more complex sequential patterns, as well as they potentially can provide more capacities to the model (Leugering et al., 2023; Moore et al., 2022). Specifically, multi-compartment models have the advantage to solve also non-separable computations, rather than only linear separable computations (Quaresima et al., 2023). Work by Quaresima et al., 2023 for example has shown that their so-called multi-compartment model, Tripod neuron, has outperformed single-compartment models in classification tasks. The authors argue that this might imply that the inclusion of a dendritic structure was beneficial. Yet, it is still open to further investigate the relevance of complex dendritic arborization in neural compuational models and their role regarding dendritic memory implementations (Leugering et al., 2023; Quaresima et al., 2023).

3 Methods

In this section, I will provide a detailed overview of the conducted experiments and their relevance to addressing the research questions. I will begin by explaining the setup utilized, including the technical configurations and the methodology employed in coding the logical dendritic trees. Furthermore, I will describe the process of generating experimental trials and I further on introduce a basic classifier developed for the experiments. Moving forward, I will provide a step-by-step account of the experimental procedure. Additionally, I will present the evaluation metrics that I chose to evaluate the performance of the classifier. Finally, I will present expected outcomes of specific experiments based on theoretical considerations and preliminary observations.

3.1 Experimental Setup

3.1.1 Design goals

To address the research question regarding the classification ability of dendritic trees with memory in the presence of timing variations, a comprehensive set of experiments was designed and conducted. To establish a testing environment, I will introduce a naive classifier that models the mechanism of plateau potentials in the context of dendritic sequence detection. The primary objective was to evaluate whether dendritic trees with memory can consistently identify sequences even when there are variations in the timing of input elements. Specifically, it was examined whether the robustness to timing variations persists across dendritic trees of varying complexity levels. Furthermore, I also decided to explore the influence of several factors, including varying plateau potential duration ($\boldsymbol{\tau}$) values, variations in standard deviations and means for input timings, diverse trial generation conditions, alphabet size, bias towards negative trials, trial count variations, and broader or narrower distributions for input timings. In using selected evaluation metrics, I evaluated the dendritic trees' ability to consistently process sequences. Through interpreting these evaluations, I attempted to infer conclusions about the classifier's robustness and therein answer the research question and validate the hypotheses.

3.1.2 Technical setup

The code for the experimental setup and plotting of figures was written in Julia, version 1.10.0. The code can be found at the following GitHub link: https://github.com/goody139/bachelor-thesis. The following packages and libraries were utilized:

- `StatsBase` - Used for sampling random sequences.
- `Distributions` - Used for sampling values from Gaussian and Uniform distributions.
- `Plots` - Utilized for creating a visualization environment to plot the performances.
- `ROCAnalysis` - Used to calculate the Area Under the Curve (AUC) score.
- `Combinatorics` - Used to generate possible combinations for dendritic trees.
- `IterTools` - Used to iterate through lists more efficiently.

3.1.3 Dendritic trees

I have decided to incorporate seven distinct dendritic trees into my experiments, as illustrated in Table 2. Since I set up the experiment with Julia, I decided to represent elements as `Symbols`. The initial dendritic tree E1 which I will also refer to as the vanilla dendritic tree, lacks logical elements and processes elements :A, :B, and :C sequentially. Consequently, the only valid combination for this structure is [:A, :B, :C]. In contrast, the subsequent dendritic trees of type D, F and G express logic in code using symbols such as ":&" for the AND operator and ":||" for the OR operator. The complexity of these structures increases from D, to F, to G. In a logical AND scenario, all elements must be true for a valid combination. For instance, in the dendritic tree D2, both [:A, :B, :C] and [:B, :A, :C] are valid combinations since the order of elements does not affect the logical outcome. Conversely, dendritic tree D1, implementing the OR operator, shares the same combinations as D1, since for a logical OR, either one or all elements could be true. Therefore, combinations such as [:B, :C] and [:A, :C] are additionally included for D1. This logic of combinations applies consistently across all other dendritic trees.

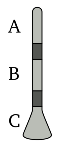
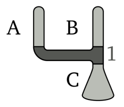
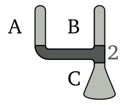
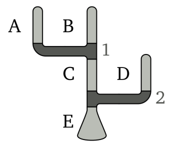
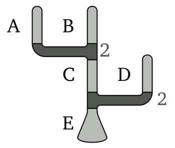
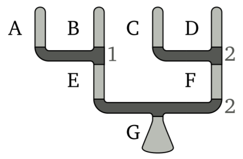
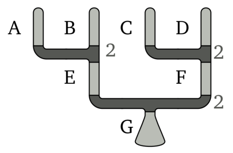
Identifier	Code	Structure
E1	[:A, :B, :C]	 $(A \rightarrow_1 B) \rightarrow_1 C$
D1	[[:||, :A, :B], :C]	 $A + B \rightarrow_1 C$
D2	[[:&, :A, :B], :C]	 $A + B \rightarrow_2 C$
F1	[[:&, [[:||, :A, :B], :C], :D], :E]	 $(A + B \rightarrow_1 C) + D \rightarrow_2 E$
F2	[[:&, [[:&, :A, :B], :C], :D], :E]	 $(A + B \rightarrow_2 C) + D \rightarrow_2 E$
G1	[[:&, [[:||, :A, :B], :E], [[:&, :D, :C], :F]], :G]	 $(A + B \rightarrow_1 E) + (C + D \rightarrow_2 F) \rightarrow_2 G$
G2	[[:&, [[:&, :A, :B], :E], [[:&, :D, :C], :F]], :G]	 $(A + B \rightarrow_2 E) + (C + D \rightarrow_2 F) \rightarrow_2 G$

Table 2: This figure presents all dendritic trees utilized in the experiments in this thesis. The columns include "Identifier," indicating how each structure will be referenced throughout the thesis; "Code," demonstrating the implementation; and "Structure," providing a visual representation. The images of the dendritic trees are sourced and adapted from (Leugering et al., 2023). To illustrate function types, a subscripted arrow ($\rightarrow$) denotes the dendritic threshold. Assuming uniform dendritic segment weights, a threshold of 1 simplifies to an OR operator, while a threshold of 2 represents a logical AND.

3.1.4 Trials

The trials that are used for classification adapt to a certain structure as can be seen in Figure 2. The number of trials is determined by a fixed trial count.

```
Tuple{Vector{Symbol}, Vector{Float64}, Bool}[(
    [:B, :A, :B, :C, :D, :C],
    [91.25630606901296,177.10744935910986,
    256.4371124144846, 330.1600846956255
    521.1600846956255, 610.25630606901296],
    1])
```

Figure 2: Structure of Experimental Trials: Sequence, Timings, and Truth Value

Namely, one trial consists of a given sequence, the associated timings for each element in the sequence and the truth value denoting whether the target sequence is part of the sequence or not (where 0 denotes false and 1 denotes true). A target sequence is part of a sequence when the target sequence appears without any other symbols in between. For example, the target sequence [:A, :B, :C] would be part of the sequence [:B, :A, :B, :C, :D, :C], whereas the target [:A, :B, :D] would not be part of the sequence since the symbol :C is in between. As Figure 2 already shows, a sequence is a Vector of Symbols. Moreover, the number of symbols that are part of a sequence is defined by a fixed sequence length. An alphabet α, which is also a Vector of symbols, defines a set of elements sequence elements are sampled from. Formally,

$$\alpha = \{a_1, a_2, \ldots, a_i\}$$

$$Sequence = \{s_1, s_2, \ldots, s_i \mid s_i \in \alpha\}$$

.

Accordingly, sequences are generated randomly from a specified alphabet - a set of symbols sequences are sampled from (α) alphabet α with a fixed sequence length. Furthermore, for each element in a sequence timings are sampled from a predefined distribution and further saved in a Vector of Floats. The timings can be interpreted as consecutive additions, wherein each symbol is associated with a specific duration. For instance, the timing for the first symbol may be 91 milliseconds, followed by subsequent symbol which sampled timing, let's say 86 milliseconds, is added to the previous duration, here 177 milliseconds. This process is outlining the temporal progression of events. Through the course of my experiments I mostly used the Gaussian distribution with Standard Deviation (σ), and Mean (μ), but also the uniform distribution with minimum values (a) and maximum values (b). It is important to note, that when sampling from the Gaussian distribution I truncated negative samples in order to prevent negative timings for the trials. I truncated the function so that if a sampled value is positive, it is returned, and if a sampled value is negative, the function is called recursively, see Figure 3.

```
function gaussian(; args::NamedTuple)
    value = rand(Normal(args.mean, args.sd))
    return value > 0 ? value : gaussian(args=(mean=args.
       mean, sd=args.sd))
end
```

Figure 3: Code for Sampling from Gaussian Distribution with Truncation of Negative Samples

In this experimental setup, trials are generated under three different conditions: *balanced, unbalanced, and extra condition.* In the balanced condition, the set of trials is composed such that 50% of the trials are correct, and the remaining 50% are false. To generate positive and negative trials, sequences are randomly generated and assigned to either the positive or negative pool until the desired number of trials is achieved. In the unbalanced condition, the set of trials is generated randomly based on a given trial count. The proportion of true to false trials is variable, without any constraints on the balance between true and false trials. The extra condition is similar to the balanced condition in that 50% of the trials are correct. However, the false trials in this condition contain the target sequence but with at least one distractor element between each element of the target sequence. For example, if the target sequence is [:A, :B, :C], a false trial in the extra condition might look like [:B, :C, **:A**, :D, **:B**, :A, :D, **:C**]. Here, the elements A, B, and C appear in the correct order but are separated by at least one other element.

3.1.5 Naive classifier

To explore the concept of plateau potentials in modeling, I devised a binary *naive classifier*[1]. The provided pseudocode, referenced as Algorithm 1, outlines the implementation of this classifier algorithm. It is designed to classify by identifying a specified target sequence within each trial, while considering the plateau potential

[1]My first supervisor devised the idea and base code for the naive classifier. I built upon his work in terms of a naive classifier for more complex dendritic trees.

duration, threshold parameter τ[2].

Algorithm 1 naive_dendritic_classifier(trials, target_sequence, τ)

```
1: $detected \leftarrow$ empty array
2: for each $trial$ in $trials$ do
3:     $target\_location \leftarrow$ find_first_target_symbols($trial$.sequence, $target\_sequence$)
4:     if $target\_location$ is not None then
5:         $detect \leftarrow$ true
6:         for $i$ from 1 to (length of $target\_location$ - 1) do
7:             $timing\_1 \leftarrow trial.timings[target\_location[i+1]]$
8:             $timing\_2 \leftarrow trial.timings[target\_location[i]]$
9:             $detect \leftarrow detect \wedge (\tau \geq (timing\_1 - timing\_2))$
10:        end for
11:        append $detected$, $detect$
12:    else
13:        append $detected$, false
14:    end if
15: end for
16: return $detected$
```

The algorithm begins by initializing an empty array called `detected` to hold the classification outcomes. Next, it iterates through each trial in the set. For every trial, it attempts to locate the initial occurrence of the target sequence using the `find_first_target_symbols` function. If the `target_location` is `None`, meaning that the `target_sequence` does not occur in the sequence, the `detect` variable is set to `false`. Upon finding the target sequence, denoted by a non-null `target_location`, the algorithm sets the `detect` variable to `true`, signifying that the `target_sequence` is part of the sequence. Subsequently, the algorithm evaluates the timing difference between consecutive elements of the target sequence. If this difference surpasses the predefined threshold τ, the detection is deemed invalid, and `detect` is switched to `false`. If τ is greater than this difference, the detection is deemed valid, and `detect` remains `true`. The outcome of this detection (`true` or `false`) is then added to the `detected` array. Lastly, the algorithm returns the `detected` array, containing the classification outcomes for each trial.

For instance, consider a target sequence `[:A, :B, :C]` with $\tau = 200$ ms and trials as depicted in Figure 2. The locations of the first target symbols would be indices `[2, 3, 4]` and the calculation (timing difference) would be as shown in Equation

[2]Note that τ is specified in milliseconds. Thus, a tau value of 200 indicates a plateau potential lasting for 200 milliseconds.

$$\begin{aligned}
&\tau \geq (timing\_B - timing\_A) \\
&= 200 \geq (256.4371124144846 - 177.10744935910986) \\
&= 200 \geq 79.32966305537474 \\
&= \text{True}
\end{aligned} \tag{1}$$

$$\begin{aligned}
&\tau \geq (timing\_C - timing\_B) \\
&= 200 \geq (330.1600846956255 - 256.4371124144846) \\
&= 200 \geq 73.7229722811409 \\
&= \text{True}
\end{aligned} \tag{2}$$

Figure 4: A demonstration on how the classification of the naive classifier works by employing an example. It is checked whether τ is surpassing the timing difference of consecutive elements. Here trials with the respective timings are taken from Figure 2, target sequence is [:A, :B, :C] and τ is 200. Equation (1) shows the calculation for symbols :A, :B, and Equation (2) the difference in between :B and :C.

1 for symbols :A and :B with indices two and three, followed by Equation 2 for symbols :B and :C, with indices three and four. It is worth noting that the target sequence `[:A, :B, :C]` might not have been detected with a τ lower than 70, as the difference between consecutive elements must surpass the tau threshold. Moreover, with a greater τ, the target sequence `[:A, :B, :D]` would be classified as correct in this example, even though the target sequence contains a distractor element, specifically `:C` in this case. In this experimental setup, a sequence is classified as true if the difference between consecutive elements of the target sequence exceeds the τ threshold. A sequence is considered 'correct' if the target sequence occurs in the given sequence without any distractors between consecutive elements. Thus, as discussed, the classifier can be described as 'naive' because it simply reads the inputs and checks them against the threshold. We observed that even when a sequence is correct, it can be classified as false if τ is lower than the temporal difference between consecutive elements. This scenario occurs in cases of large temporal distances or low τ thresholds.

It is also important to highlight that the 'naive' classifier encounters edge cases that are not handled, further underscoring its simplicity. For instance, in a sequence such as `[:A, :A, :B, :C]`, the target sequence `[:A, :B, :C]` could be detected correctly. However, the classifier would detect the sequence starting from the first `:A` instead of the second `:A`. While this detection is technically correct, it is an undesirable outcome that requires proper handling. Similarly, for sequences with repeated elements, such as `[:A, :B, :B, :C]`, the classifier could still detect `[:A, :B, :C]` if τ is sufficiently large and the differences between elements are small enough. For a detailed discussion on the impact of edge cases on the classifier's performance, see Section 5.

Algorithm 2 naive_dendritic_classifier_branchings(trials, target_sequence, τ)

```
1: detected ← empty array
2: for each trial in trials do
3:     target_locations ← find_first_target_symbols(trial.sequence, target_sequence)
4:     if any($x \to$ test_target_location($x$, trial, $\tau$), target_locations) then
5:         push!(detected, true)
6:     else
7:         push!(detected, false)
8:     end if
9: end for
```

For now I explained the basic functionality of the naive classifier that can handle simple target sequences like E1. However, for more complex logical dendritic trees, this approach must be adapted to accommodate the multiple possible combinations, as elaborated in Section 3.1.3. The classifier for these logical dendritic trees is presented in pseudocode in Algorithm 2. Initially, the procedure is similar to the naive classifier: a variable `detected` is initialized as an empty array. For each trial in trials, the function `find_first_target_symbols` is called with the trial's sequence and the target sequence. This function identifies all possible combinations for the target sequence and saves them in a list. It then returns a list of indices corresponding to the elements of the target sequence that first appear in the trial sequence, storing these indices in the variable `target_locations`. Next, for each target location, the function `test_target_location` checks whether the τ threshold is surpassed, i.e., whether the target sequence is detected. This function implements the threshold method as in the naive classifier, but for simplicity, it is encapsulated in a separate function. Based on the outcome of testing the target locations, true or false is appended to the detected variable, following the same procedure as in the naive classifier.

Since this approach is based on the naive classifier it has the same undesired outcomes, but one dimension is changed due to multiple possible combinations. For example, consider the target sequence "D1" with possible combinations "BAC" and "ABC". The current implementation classifies a sequence as true if any combination returns true from the any function. For instance, ABC might be present in a sequence but fail the τ threshold check, while BAC, not in direct sequence, might incorrectly pass the check. This results in a technically correct classification but for the wrong reason.

3.2 Experiments

3.2.1 Procedure

First of all, I conducted one main experiment for the dendritic trees of type E, D, F and G in order to explore the robustness of sequence detection to timing jitters for dendritic trees with varying logical complexity. The main experiment for a single dendritic tree was structured as follows. The experiment was executed separately with three different standard deviations (10, 25, and 50). For each standard deviation σ, the experiment was repeated 400 times, varying the mean μ of the Gaussian distribution from 1 to 400. During each iteration, trials were generated, with conditions being either balanced, unbalanced, or following an extra condition. The total trial count was set to 600. The sequence lengths were adapted to the length of the dendritic trees, specifically choosing a length of $2\times$ length of longest combination $+4$. Consequently, for structures E1, D1, and D2, the sequence length was 10; for structures F1 and F2, it was 14; and for the G structure, it was 18. The alphabet was determined based on all the elements that are part of the dendritic tree plus one additional element. For example, the alphabet for structures E1, D1, and D2 was {:A, :B, :C, :D}, for the F structures it was {:A, :B, :C, :D, :E, :F}, and for the G structure, it was {:A, :B, :C, :D, :E, :F, :G, :H}. The classifier was applied across a range of τ values from 20 to 600. Predictions and performance metrics were recorded for each combination of mean and standard deviation. Therefore, the performance data comprised a 400-element vector, where for every mean value and standard deviation combination, every τ parameter from 20 to 600 was tested. A visual representation of the performance structure is shown in Figure 5. This means that the result of one main experiment for one dendritic tree comprises 9 performance structures: 3 different trial generation conditions combined with 3 different standard deviations. In total, this yields for all 7 dendritic trees 63 performance results.

In addition, I conducted four supplementary experiments focusing on the dendritic tree E1. Firstly, I explored the impact of varying trial counts by conducting the main experiment with three different trial counts. Secondly, I investigated the influence of introducing a bias towards negative trials by augmenting the number of negative trials under balanced conditions. Thirdly, I examined whether the amount

```
400-element Vector{Tuple}:
[((mean = 1, sd = 10), [(20.0, 1, 2, 3, 4), ... (600.0, 5,
   0, 5, 0)])
...
 ((mean = 400, sd = 10), [(20.0, 0, 5, 0, 5), ... (600.0,
   0, 4, 1, 5)])]
```

Figure 5: Illustration of an exemplary performance structure resulting from one experiment. The experiment involved a σ of 10, μ ranging from 1 to 400, and τ values spanning from 20 to 600 for each standard deviation and mean combination. Each parameter combination underwent 10 trials for a single target sequence. The structure is presented as a 400-element vector of tuples, where each tuple contains the parameters (mean, sd) and a list of performance metrics (τ, FP, FN, TP, TN), where False Positive (FP), False Negative (FN), True Positive (TP), and True Negative (TN).

of elements comprising an alphabet has an effect on performance by conducting the main experiment with three different alphabets. Lastly, I examined whether broader or narrower distributions could elicit different effects on performance for a fixed τ value of 100. To assess this, I employed the uniform distribution and tested wider and narrower ranges.

3.2.2 Evaluation metrics

In order to evaluate the performance of the binary classification system, the evaluation metrics Accuracy, F-score, Sensitivity, Specificity, Matthew Correlation Coefficient (MCC), Receiver Operating Characteristic (ROC), and connected to that the the Area under curve (AUC), are used. Specifically, they are suitable to evaluate performances of binary classification systems as elaborated in (Akosa, 2017; Fawcett, 2006; Hicks et al., 2021). These variety of measures might be able to depict and provide an interpretation basis.

Accuracy

The accuracy provides an overall assessment of the system's performance. It indicates the proportion of correctly classified instances out of all instances (Hicks et al., 2021).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{3}$$

Sensitivity

Sensitivity measures the system's ability to correctly identify true positive instances of sequence detection, indicating how well it captures the presence of sequences within the data (Hicks et al., 2021).

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4}$$

Specificity

Specificity measures the system's ability to correctly identify true negative instances, indicating its capacity to avoid incorrectly labeling non-sequence data as sequences (Hicks et al., 2021).

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{5}$$

F-score

The F-score combines precision and recall (sensitivity) into a single metric, offering a balanced evaluation of the system's ability to detect sequences while considering FP and FN (Hicks et al., 2021).

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

Balanced Accuracy

Balanced accuracy provides a more accurate measure of model performance when classes are imbalanced because it considers both the sensitivity and specificity for each class, rather than just overall accuracy, which can be misleading when classes are unevenly distributed (Akosa, 2017). Sensitivity and Specificity are defined as above. In the context of my experiments I therefore used this measure to evaluate performances that were based on the unbalanced trial generation.

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity } + \text{ Specificity}}{2} \tag{7}$$

ROC curve & AUC

The ROC curve assesses the system's ability to discriminate between positive and negative instances across different threshold settings, offering a comprehensive evaluation of its discriminative power (Fawcett, 2006). Connected to the ROC curve it is usual to also compute the AUC measure which is representing a portion of the unit square's area, always ranges between 0 and 1 (Fawcett, 2006). However, as random guessing yields the diagonal line between (0, 0) and (1, 1) with an area of 0.5, any practical classifier should surpass an AUC of 0.5 (Fawcett, 2006). As depicted below the AUC is calculated by integrating the True Positive Rate (TPR) over the range of False Positive Rate (FPR) from 0 to 1. The True positive rate equals to the sensitivity score as introduced above, and the False positive rate is introduced below. The ROC curve is plotted in a graph by plotting TPR against FPR.

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{8}$$

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{9}$$

MCC

MCC acts as a correlation metric between true and predicted classes, offering insights across all confusion matrix entries (Hicks et al., 2021). Ranging from -1 to 1, MCC achieves high scores when predictions are accurate; perfect prediction is denoted by 1, random guessing by 0, and complete disagreement by -1 (Hicks et al., 2021).

$$\text{MCC} = \frac{\text{TP * TN - FP * FN}}{\sqrt{(\text{TP} + \text{FP}) * (\text{TP} + \text{FN}) * (\text{TN} + \text{FP}) * (\text{TN} + \text{FN})}} \tag{10}$$

3.2.3 Expected Results

1) E1 for 3sd –> performance for unbalanced condition worse (more difficult setup) (performance and also F1 and MMC should fall bc TN should be way lower) hypothesis: despite varying timings still be able to detect

i expect an optimal tau value range (thoughts: for too low tau not able to classify and for too large too many false trials that are fälschlicherweise detected (idee: da wo es drin ist, ist tau immer groß genug um es zu classifien, da wo es zu klein ist erkennt er sie nicht)) –> result : für trials wo sequence nicht in der Reihenfolge drin ist wird direkt als falsch classified und tau wird nicht in betracht gezogen bei balanced wo einfach nur falsche trials drin sind ist da vermutlich der Anteil hoch und deshalb bleibt es trotzdem so hoch. Extra condition, die NUR diese beispiele drin hat hat dann gezeigt, dass es eine optimum tendenz gibt (F1 score saturates bc TP are 100

% and score is based on TP. Therefore the MCC considering every score could show an optimal behavior here.

I expect that

I hypothesize that ... I aim to test the hypotheses that the classifier will detect robustly throughout complexity of branches with variations in timings. test whethere there is an optimal tau.

Based on initial observations (edge cases, conditions)

since extra condition forces to more difficult false trials and not handled edge cases I suppose it will perform worse

4 Results

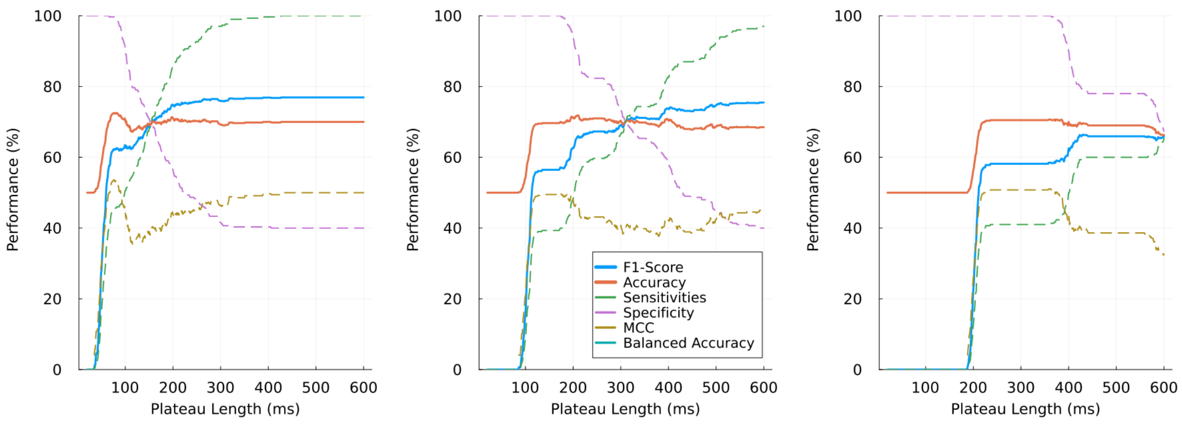
adding plots of results here

4.1 Vanilla branching structure

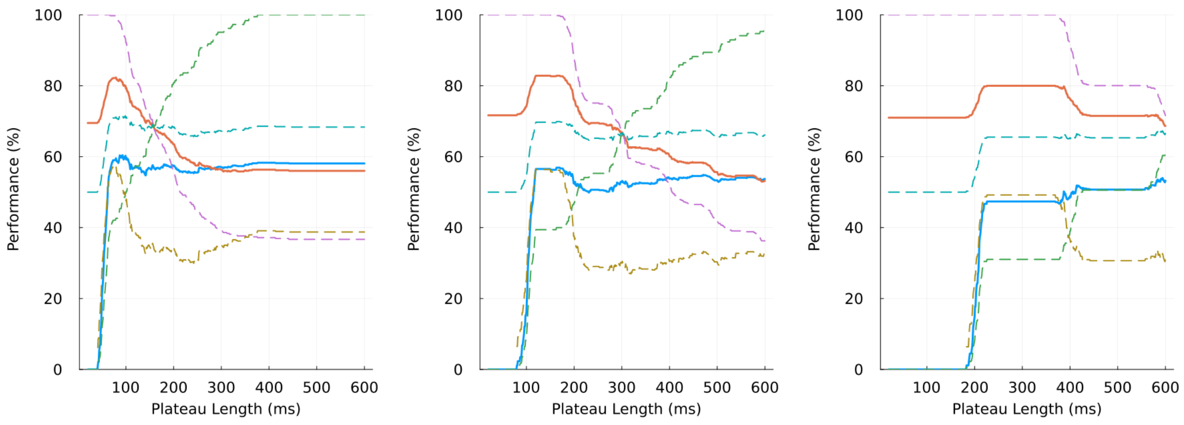
(London & Häusser, 2005) (Antic et al., 2010) (Bouhadjar et al., 2022)

4.2 Simple branching structures

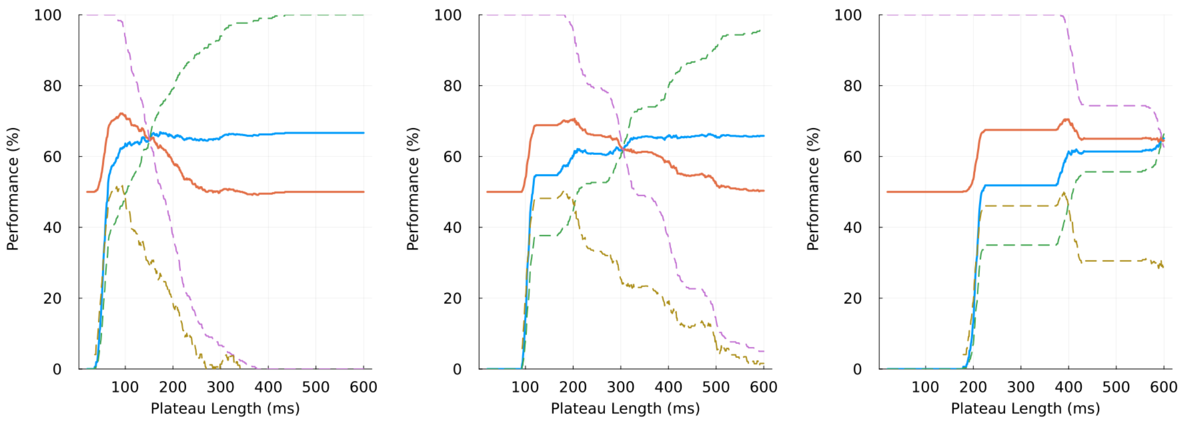
4.3 Side Experiments



(a) balanced

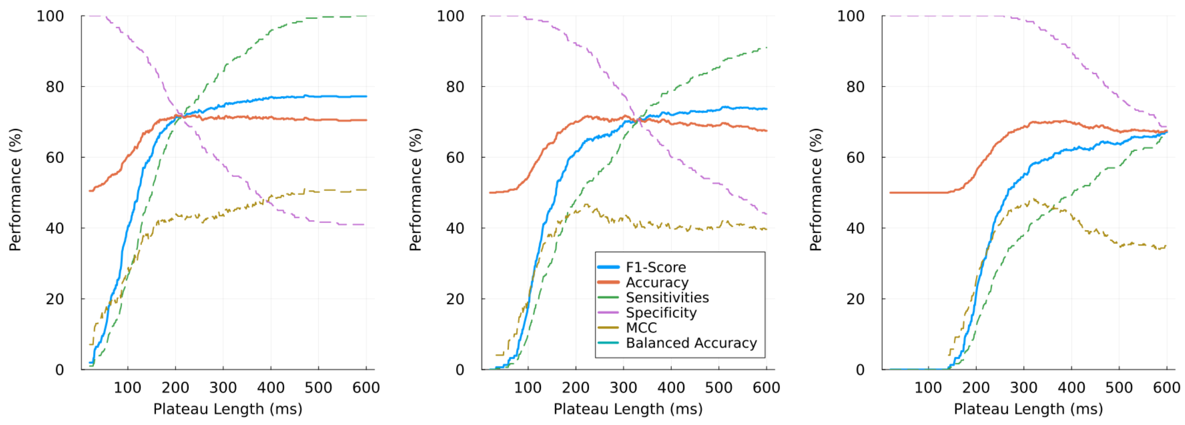


(b) unbalanced

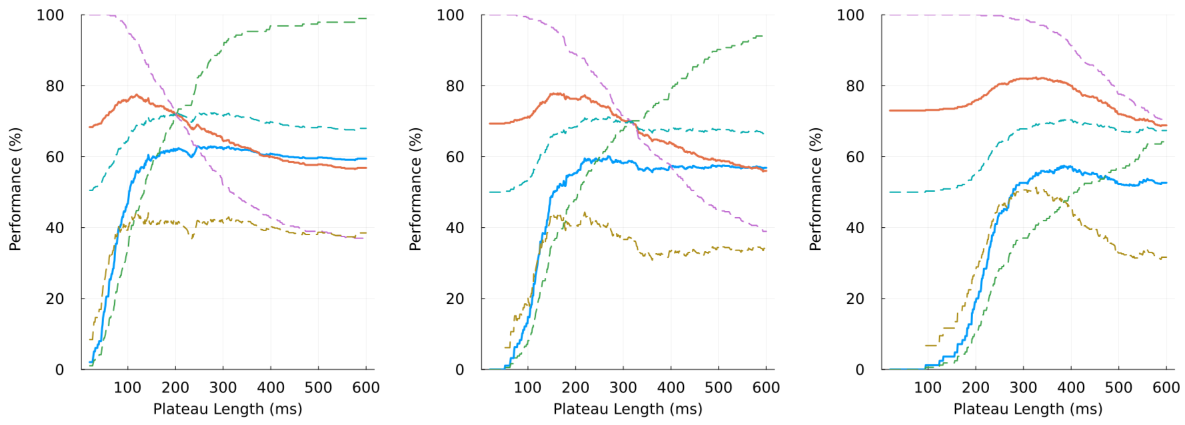


(c) extra condition

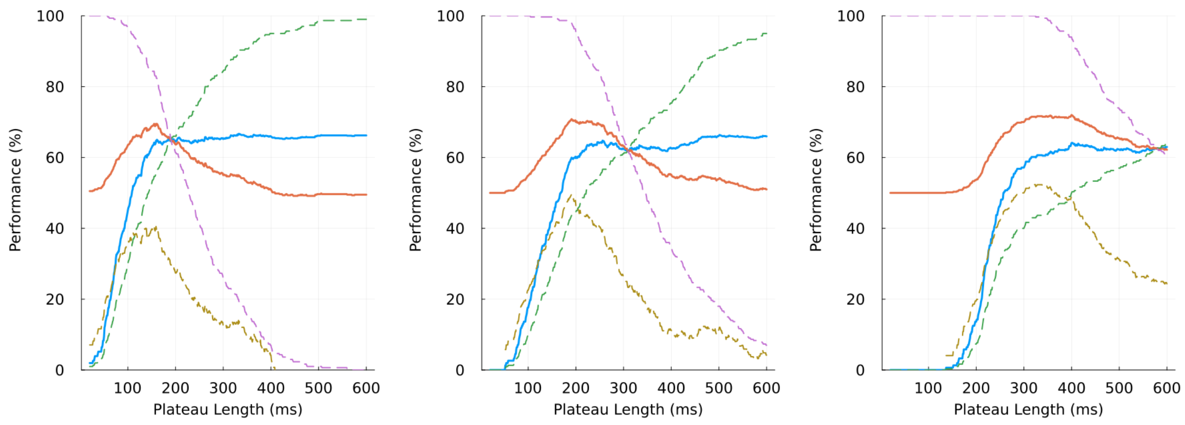
Figure 6: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify the target E1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values of 50, 100, and 200, spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced



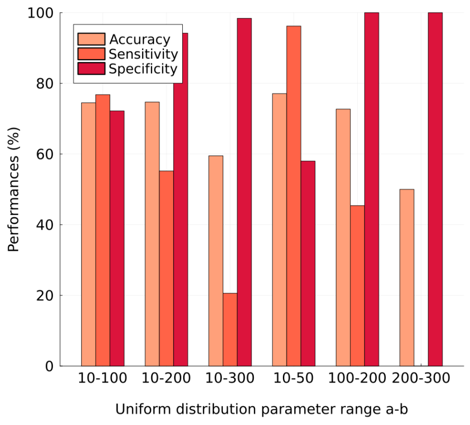
(b) unbalanced



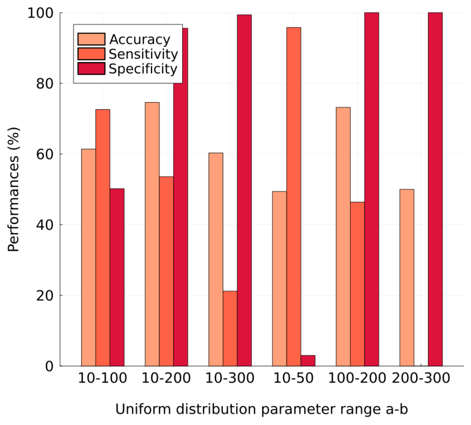
(c) extra condition

Figure 7: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify the target E1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values of 50, 100, and 200, spatially depicted as 50 (left), 100 (middle), and 200 (right).

[1]Notably, Balanced Accuracy is exclusively applied to the unbalanced condition.

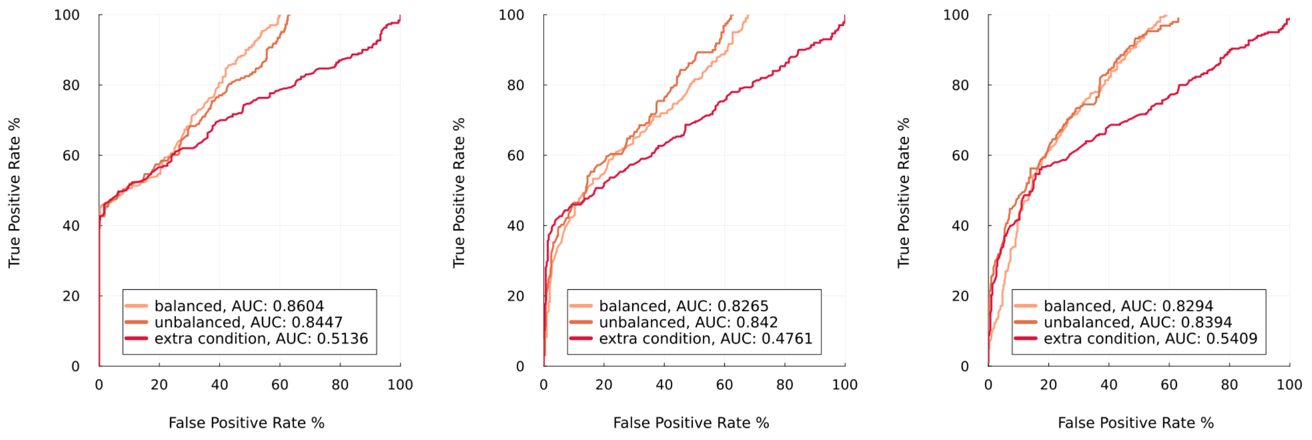


(a) Balanced

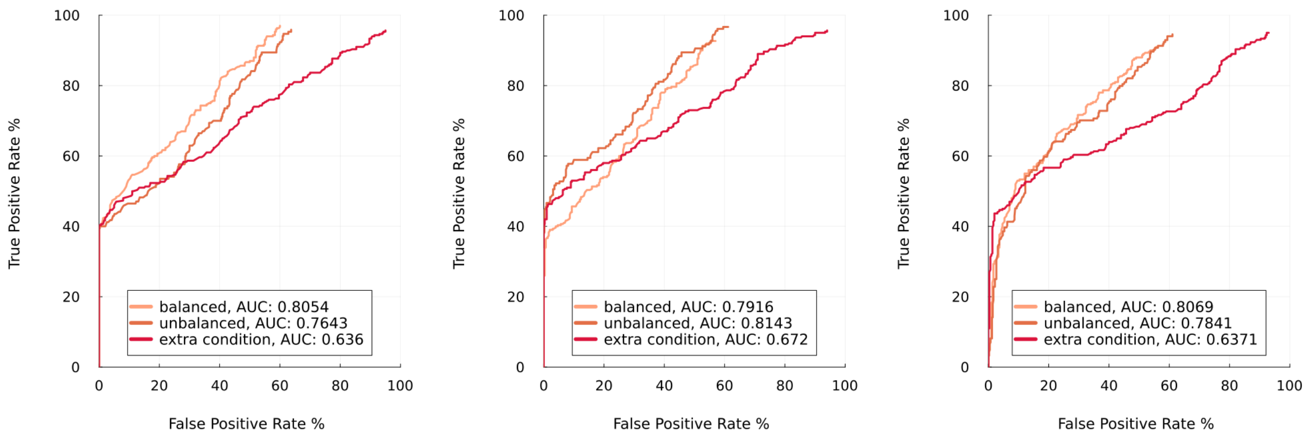


(b) Extra condition

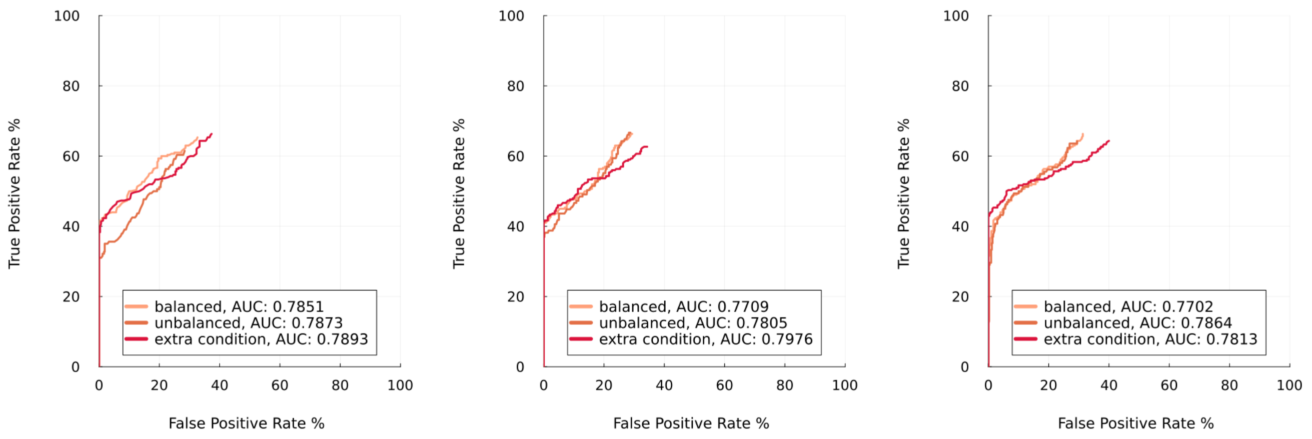
Figure 8: The figure illustrates the performances (Accuracy, Sensitivity, Specificity) of the naive classifier trying to classify dendritic tree E1 with conditions a) Balanced and b) Extra condition. It involved 1000 trials with a τ of 200 and a sequence length of 10. Trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Uniform distribution with an a and b ranging as indicated in the figure.



(a) $\mu = 50$



(b) $\mu = 100$



(c) $\mu = 200$

Figure 9: The figure illustrates the ROC curves and AUC scores of the naive classifier in its attempt to classify the target E1 across balanced, unbalanced, and extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 10, 25, 50 spatially depicted as 10 (left), 25 (middle), and 50 (right) and μ values of a) 50, b) 100, and c) 200.

5 Discussion

5.1 Interpret results

do results answer research question and confirm initial hypotheses?

5.2 Limitations

what is not considered?

5.3 Outlook

what would be interesting to know for future work? How are my experiments fitting into current research and literature?

6 Conclusion

7 Acknowledgments

First and foremost, I would like to thank...

- advisers
- examiner
- person1 for the dataset
- person2 for the great suggestion
- proofreaders

Bibliography

Akosa, J. S. (2017). Predictive accuracy : A misleading performance measure for highly imbalanced data. https://api.semanticscholar.org/CorpusID:43504747

Antic, S. D., Zhou, W.-L., Moore, A. R., Short, S. M., & Ikonomu, K. D. (2010). The decade of the dendritic nmda spike. *Journal of Neuroscience Research*, *88*(14), 2991–3001. https://doi.org/https://doi.org/10.1002/jnr.22444

Augustinaite, S., Kuhn, B., Helm, P. J., & Heggelund, P. (2014). Nmda spike/-plateau potentials in dendrites of thalamocortical neurons. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, *34*(33), 10892–10905. https://doi.org/10.1523/JNEUROSCI.1205-13.2014

Billaudelle, S., & Ahmad, S. (2016). Porting htm models to the heidelberg neuromorphic computing platform.

Bouhadjar, Y., Wouters, D. J., Diesmann, M., & Tetzlaff, T. (2022). Sequence learning, prediction, and replay in networks of spiking neurons. *PLOS Computational Biology*, *18*(6), 1–36. https://doi.org/10.1371/journal.pcbi.1010233

Branco, T., Clark, B. A., & Häusser, M. (2010). Dendritic discrimination of temporal input sequences in cortical neurons. *Science*, *329*(5999), 1671–1675. https://doi.org/10.1126/science.1189664

Burger, T. S., Rule, M. E., & O'Leary, T. (2023). Active dendrites enable robust spiking computations despite timing jitter. https://doi.org/10.1101/2023.03.22.533815

Cardwell, S. G., & Chance, F. S. (2023). Dendritic computation for neuromorphic applications. *Proceedings of the 2023 International Conference on Neuromorphic Systems*. https://doi.org/10.1145/3589737.3606001

Cuntz, H., Remme, M., & Torben-Nielsen, B. (2014). *The computing dendrite* (Vol. 11). Springer. https://doi.org/10.1007/978-1-4614-8094-5

D'Agostino, S., Moro, F., Torchet, T., Demirag, Y., Grenouillet, L., Indiveri, G., Vianello, E., & Payvand, M. (2023). Denram: Neuromorphic dendritic architecture with rram for efficient temporal processing with delays.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.-K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., . . . Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, *38*(1), 82–99. https://doi.org/10.1109/MM.2018.112130359

Fawcett, T. (2006). An introduction to roc analysis [ROC Analysis in Pattern Recognition]. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/https://doi.org/10.1016/j.patrec.2005.10.010

Ferrante, M., Migliore, M., & Ascoli, G. (2013). Functional impact of dendritic branch-point morphology. *J Neurosci*, *33*(5), 2156–2165. https://doi.org/10.1523/JNEUROSCI.3495-12.2013

Gao, P. P., Graham, J. W., Zhou, W.-L., Jang, J., Angulo, S., Dura-Bernal, S., Hines, M., Lytton, W. W., & Antic, S. D. (2021). Local glutamate-mediated dendritic plateau potentials change the state of the cortical pyramidal neuron [PMID: 33085562]. *Journal of Neurophysiology*, *125*(1), 23–42. https://doi.org/10.1152/jn.00734.2019

Gasparini, S., Migliore, M., & Magee, J. C. (2004). On the initiation and propagation of dendritic spikes in ca1 pyramidal neurons. *Journal of Neuroscience*, *24*(49), 11046–11056. https://doi.org/10.1523/JNEUROSCI.2520-04.2004

Hammouamri, I., Khalfaoui-Hassani, I., & Masquelier, T. (2023). Learning delays in spiking neural networks using dilated convolutions with learnable spacings.

Hawkins, J., & Ahmad, S. (2016). Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, *10*. https://doi.org/10.3389/fncir.2016.00023

Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2021). On evaluation metrics for medical applications of artificial intelligence. *medRxiv*. https://doi.org/10.1101/2021.04.07.21254975

Kaiser, J., Billaudelle, S., Müller, E., Tetzlaff, C., Schemmel, J., & Schmitt, S. (2022). Emulating dendritic computing paradigms on analog neuromorphic hardware [Dendritic contributions to biological and artificial computations]. *Neuroscience*, *489*, 290–300. https://doi.org/https://doi.org/10.1016/j.neuroscience.2021.08.013

Leugering, J., Nieters, P., & Pipa, G. (2023). Dendritic plateau potentials can process spike sequences across multiple time-scales. *Frontiers in Cognition*, *2*. https://doi.org/10.3389/fcogn.2023.1044216

London, M., & Häusser, M. (2005). Dendritic computation [PMID: 16033324]. *Annual Review of Neuroscience*, *28*(1), 503–532. https://doi.org/10.1146/annurev.neuro.28.061604.135703

Major, G., Polsky, A., Denk, W., Schiller, J., & Tank, D. W. (2008). Spatiotemporally graded nmda spike/plateau potentials in basal dendrites of neocortical pyramidal neurons [PMID: 18337370]. *Journal of Neurophysiology*, *99*(5), 2584–2601. https://doi.org/10.1152/jn.00011.2008

Milojkovic, B. A., Radojicic, M. S., Goldman-Rakic, P. S., & Antic, S. D. (2004). Burst generation in rat pyramidal neurones by regenerative potentials elicited in a restricted part of the basilar dendritic tree. *The Journal of Physiology*, *558*(1), 193–211. https://doi.org/https://doi.org/10.1113/jphysiol.2004.061416

Moore, J., Robert, V., Rashid, S., & Basu, J. (2022). Assessing local and branch-specific activity in dendrites [Epub 2021 Oct 29]. *Neuroscience*, *489*, 143–164. https://doi.org/10.1016/j.neuroscience.2021.10.022

Moradi, S., Qiao, N., Stefanini, F., & Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Transactions on Biomedical Circuits and Systems*, *12*(1), 106–122. https://doi.org/10.1109/tbcas.2017.2759700

Nevian, T., Larkum, M. E., Polsky, A., & Schiller, J. (2007). Properties of basal dendrites of layer 5 pyramidal neurons: A direct patch-clamp recording study. *Nature Neuroscience*, *10*(2), 206–214. https://doi.org/10.1038/nn1826

Quaresima, A., Fitz, H., Duarte, R., Broek, D. v. d., Hagoort, P., & Petersson, K. M. (2023). The tripod neuron: A minimal structural reduction of the dendritic tree. *The Journal of Physiology*, *601*(15), 3265–3295. https://doi.org/https://doi.org/10.1113/JP283399

Rhodes, P. (2006). The properties and implications of nmda spikes in neocortical pyramidal cells. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, *26*(25), 6704–6715. https://doi.org/10.1523/JNEUROSCI.3791-05.2006

Schiller, J., Major, G., Koester, H., & Schiller, Y. (2000). Nmda spikes in basal dendrites of cortical pyramidal neurons. *Nature*, *404*, 285–9. https://doi.org/10.1038/35005094

Sinha, M., & Narayanan, R. (2022). Active dendrites and local field potentials: Biophysical mechanisms and computational explorations [Dendritic contributions to biological and artificial computations]. *Neuroscience*, *489*, 111–142. https://doi.org/https://doi.org/10.1016/j.neuroscience.2021.08.035

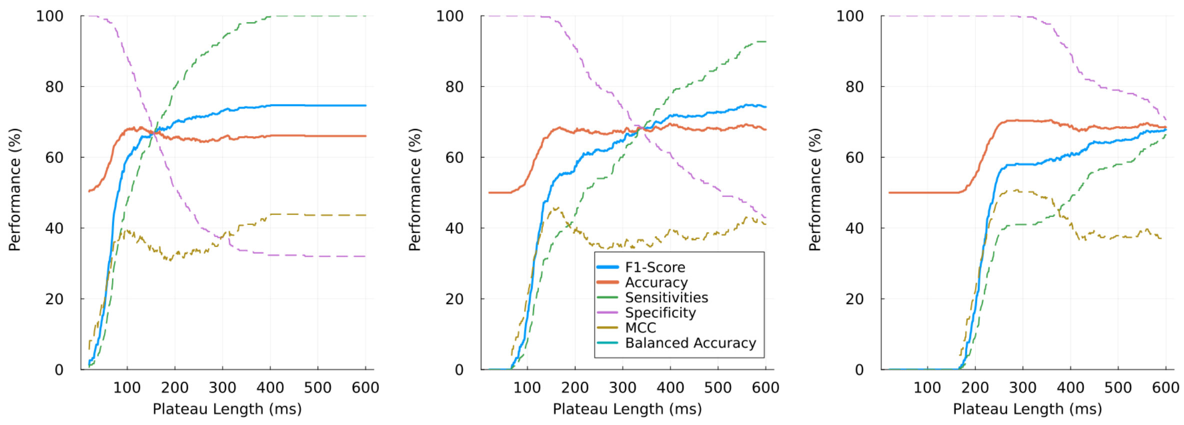
Stuart, G., Spruston, N., & Häusser, M. (2016). *Dendrites*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198745273.001.0001

Suzuki, T., Kodama, S., Hoshino, C., Izumi, T., & Miyakawa, H. (2008). A plateau potential mediated by the activation of extrasynaptic nmda receptors in rat hippocampal ca1 pyramidal neurons. *European Journal of Neuroscience*, *28*(3), 521–534. https://doi.org/https://doi.org/10.1111/j.1460-9568.2008.06324.x

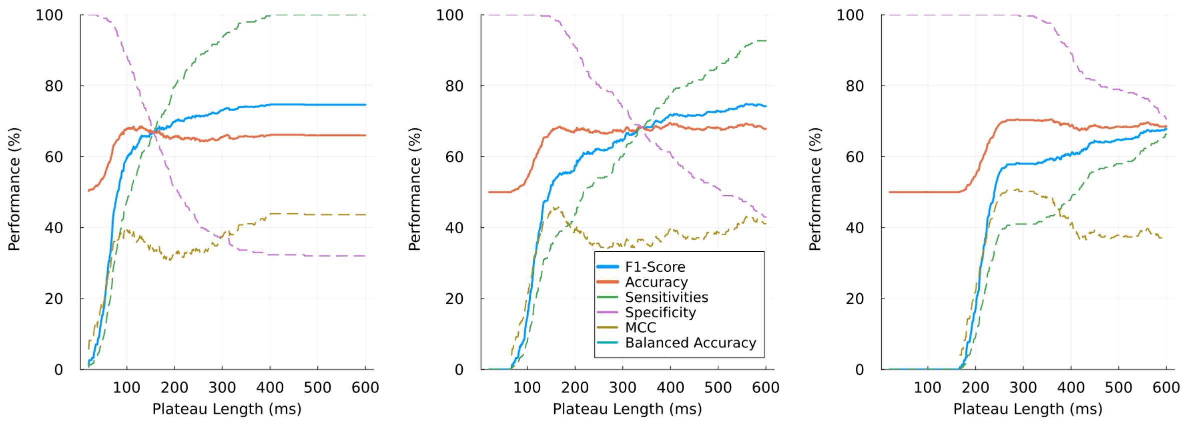
Yang, S., Gao, T., Wang, J., Deng, B., Lansdell, B., & Linares-Barranco, B. (2021). Efficient spike-driven learning with dendritic event-based processing. *Frontiers in Neuroscience*, *15*. https://doi.org/10.3389/fnins.2021.601109

Appendix

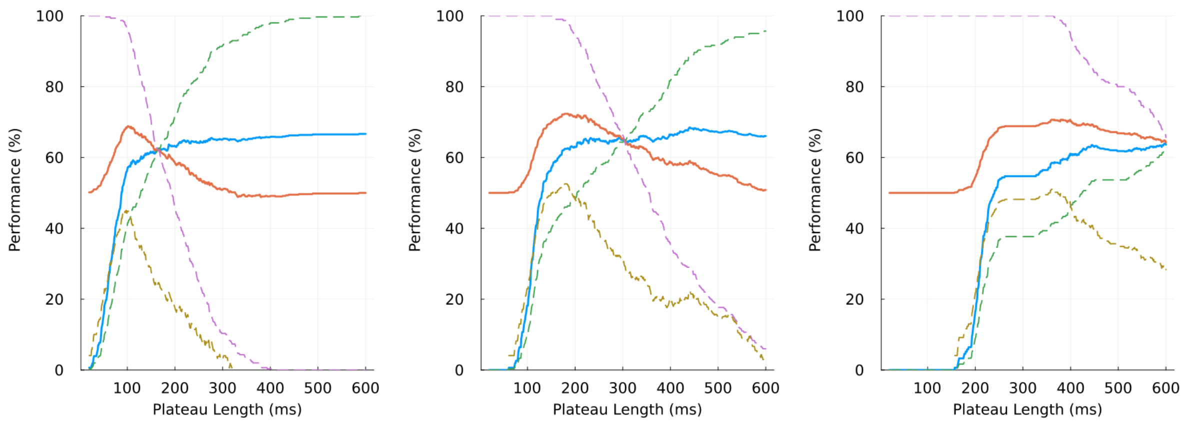
Additional figures



(a) balanced



(b) unbalanced



(c) extra condition

Figure 10: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify the target E1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values of 50, 100, and 200, spatially depicted as 50 (left), 100 (middle), and 200 (right).