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 14th, 2024

Writing period

07. 03. 2024 – 14. 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.

Osnabrück, 14.6.2024

Place, Date

Signature

Abstract

In recent years, dendritic sequence processing has gained increasing interest within computational neuroscience motivated by its potential to unravel neural computation, particularly in sequence detection and the creation of efficient systems for sequence detection and neuromorphic hardware design. This study investigates the robustness of Dendritic Trees with memory in classifying sequences under timing variations through empirical experimentation and computational modeling. Findings confirm the robustness conferred by plateau potentials and temporal overlaps, crucial to dendritic sequence processing. Evaluation metrics across varied Gaussian distribution parameters mean and standard deviation combinations support the system's effectiveness, achieving 70-80% accuracy for the extra trial generation condition with an optimal plateau potential duration τ. Factors such as the sequence pool α, negative bias, and input timing dispersion influence system robustness. Limitations include the use of a simplified model and choice of input timing distribution. Future research should explore biorealistic models and alternative distributions, while investigating complex arborization's role and learning mechanisms for enhanced understanding of dendritic sequence processing.

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

μ Mean

σ Standard Deviation

a Minimum values

b Maximum values

$\boldsymbol{\tau}$ Plateau Potential duration

1 Introduction

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 time frames is an essential 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

Specifically, the mechanisms underlying plateau potentials in active dendrites, as discussed in Section 2.1.2, involve the ability to perform time-invariant sequence detection (Branco et al., 2010; Gasparini et al., 2004; Leugering et al., 2023; London & Häusser, 2005) and are believed to play a crucial role in this process. Plateau potential-based sequence detection finds applications in various domains where robust temporal processing is essential. One such application is in natural language understanding. We can comprehend utterances across a wide range of speech tempos and rhythms, as long as the order of specific sounds is maintained (Leugering et al., 2023). For instance, in

the auditory cortex, sequences of brief phonemes encode longer syllables, which can last for around 200 ms (Leugering et al., 2023). Plateau potential-based sequence detection mechanisms enable the identification and processing of these sequential patterns in speech (Leugering et al., 2023). Additionally, in the field of neuroscience research, modeling of plateau potential-based sequence detection can shed light on the underlying mechanisms of neural information processing. By studying how dendrites detect and process sequences of synaptic activations, researchers can gain insights into fundamental cognitive processes such as memory formation, learning, and decision-making (Gao et al., 2021; Rhodes, 2006; Schiller et al., 2000). Furthermore, plateau computation facilitates the development of more efficient neuromorphic hardware systems (Bouhadjar et al., 2022; D'Agostino et al., 2023; Leugering et al., 2023). The utilization of dendritic sequence processing principles in hardware design enables the creation of neuromorphic systems that can efficiently mimic the temporal processing capabilities of the brain (Cardwell & Chance, 2023). Such advancements in neuromorphic hardware are promising as discussed more detailed in Section 2.2.2.

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.

Research question & Hypotheses

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, through empirical investigation and computational modeling 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 Dendritic Trees with significant variations in input timings, 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 incorporating simpler and more complex logic can classify sequences robustly to timing jitters indicating the importance of considering dendritic morphology in optimizing sequence detection mechanisms. Additionally, I postulate that there

exists a correlation between the length of the plateau and robust time-invariant sequence detection, with an optimal range falling between 200-600 milliseconds (ms), as suggested by existing literature (see literature review in Section 2.1.4).

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 Chapter 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 is provided regarding the experiment itself like the evaluation metrics, parameters used as well as a detailed description of the procedure. Afterwards, the Chapter 4 presents the results of the experiments and with given figures the data is analyzed and trends are identified. In the discussion Chapter 5 then 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:

- Providing a comprehensive overview of the background knowledge concerning dendritic sequence processing, integrating neurobiological insights and theoretical computational research.
- Introducing a simplified model environment, including modeled Dendritic Tree structures and a naive classifier based on the plateau potential mechanism.
- Addressing the question of whether temporal overlap sequence detection mechanisms confer robustness despite timing variations for both simple and more complex Dendritic Trees. Additionally, highlighting the system's performance with an optimal plateau potential duration range and identifying factors contributing to its robustness.

- Offering critical reflections and posing further questions to guide future research in this area.

2 Background

The current state of research concerning dendritic computing needs to be considered as multi-faceted. Firstly, I will provide current neurobiological knowledge about how dendrites work, what they are capable of, the dendritic spikes they propagate and how they are connected to sequence processing. Further, I will focus on the phenomenon of so-called plateau potentials induced in dendrites, their duration and I will discuss dendritic arborizations. Secondly, I will transition to the theoretical computational aspect, discussing state-of-the-art coincidence detection systems, their integration into neuromorphic hardware, and modeling aspects like plateau potential duration and dendritic Branching Structures in computational frameworks.

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, which support 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

Summarizing 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)

Antic et al., 2010 further describes the second decade, 2000-2009 that focused on thin branches in Basal, Oblique and Tuft regions. An interesting aspect 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 (Tetrodotoxin and Cadmium-Ion), 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 Na^+ and Ca^{2+} 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 Ca^{2+} 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 (Antic et al., 2010). They exert a local effect within the Dendritic Tree, influencing nearby synaptic inputs and contributing to prolonged excitation and impacting neuronal firing patterns and network activity (Cuntz et al., 2014).

In literature there are multiple 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. 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 outlined previously, 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 time-invaiant sequence detection (Branco et al., 2010; Gasparini et al., 2004; Leugering et al., 2023; London & Häusser, 2005).

Specifically, summarizing work by Cuntz et al., 2014, dendrites act as delay lines, causing synapses activated along them in different directions to elicit varied responses at the soma. 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 rather than a pure current source. As argued by Cuntz et. al. (2014) this suggests that dendrites are well-suited for temporal order detection,

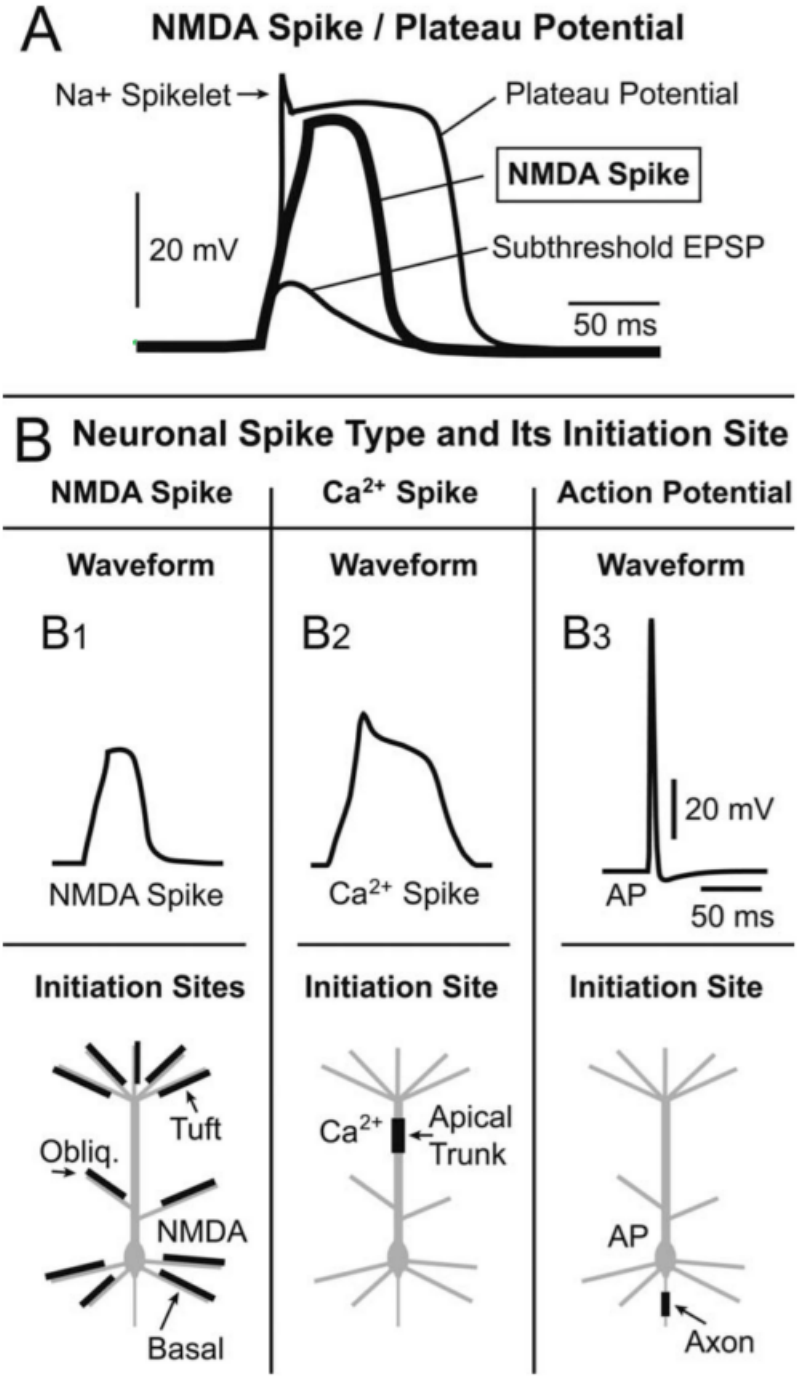


Figure 1: A comparison between dendritic spikes ('pure' NMDA spike, plateau potential, Ca^{2+} spike, and action potential) regarding waveform and initiation site. The figure is token from Antic et al., 2010.

whereas the mechanism of the plateau potential facilitate this ability. (Cuntz et al., 2014)

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. (London & Häusser, 2005)

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. Work by 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. (Leugering et al., 2023)

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

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 measured half-width.

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. 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 ms, as well as reported factors influencing the duration of the plateau.

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 ms. 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. Work by 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 have also analyzed 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.

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 (Stuart et al., 2016). Furthermore, more intricate branching patterns can enable individual neurons to detect more complex sequential patterns rendering them an interesting subject for research (Leugering et al., 2023).

2.2 Theoretical research

2.2.1 Coincidence detection systems

The detection of local features (sequences) across various time scales and spatial alignment, 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, exhibit a high demand for energy and need to be finely tuned (D'Agostino et al., 2023). Therefore, research focused on passive dendrite dynamics which are useful for pattern detection (D'Agostino et al., 2023) . 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, 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. 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; 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 (Bouhadjar et al., 2022). 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 a plateau potential 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 proceed with a basic classifier developed for the experiments. Moving forward, I will provide a step-by-step account of the experimental procedure and 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 & StatsPlots` - 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. Throughout this work, I will use the terms Dendritic Trees and Branching Structures interchangeably.

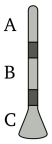
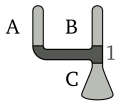
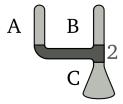
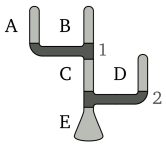
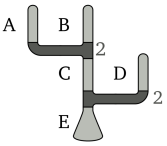
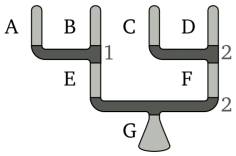
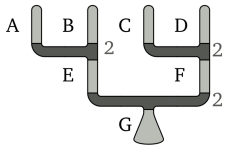
Identifier	Code	Structure
E1	[:A, :B, :C]	 $(A \to_1 B) \to_1 C$
D1	[[:||, :A, :B], :C]	 $A + B \to_1 C$
D2	[[:&, :A, :B], :C]	 $A + B \to_2 C$
F1	[[:&, [[:||, :A, :B], :C], :D], :E]	 $(A + B \to_1 C) + D \to_2 E$
F2	[[:&, [[:&, :A, :B], :C], :D], :E]	 $(A + B \to_2 C) + D \to_2 E$
G1	[[:&, [[:||, :A, :B], :E], [[:&, :D, :C], :F]], :G]	 $(A + B \to_1 E) + (C + D \to_2 F) \to_2 G$
G2	[[:&, [[:&, :A, :B], :E], [[:&, :D, :C], :F]], :G]	 $(A + B \to_2 E) + (C + D \to_2 F) \to_2 G$

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 ($\to$) 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:

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

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

Accordingly, sequences are generated randomly from a specified Alphabet (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 duration, threshold parameter τ[2].

[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.

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

Algorithm 1 Pseudocode of the naive dendritic classifier

```
1: procedure NAIVE_DENDRITIC_CLASSIFIER(trials, target_sequence, τ)
2:     detected ← empty array
3:     for each trial in trials do
4:         target_location ← find_first_target_symbols(trial.sequence, target_sequence)
5:         if target_location is not None then
6:             detect ← true
7:             for i from 1 to (length of target_location - 1) do
8:                 timing_1 ← trial.timings[target_location[i + 1]]
9:                 timing_2 ← trial.timings[target_location[i]]
10:                detect ← detect ∧ (τ ≥ (timing_1 − timing_2))
11:            end for
12:            append detected, detect
13:        else
14:            append detected, false
15:        end if
16:    end for
17:    return detected
18: end procedure
```

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 1 for symbols :A and :B with indices two and three, followed by Equation 2 for symbols :B

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

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

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.

and :C, with indices three and four. It is worth noting that the target sequence `[:A, :B, :C]` would not be 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 Pseudocode of the naive dendritic classifier for logical Dendritic Trees

```
1:  procedure CLASSIFIER_BRANCHINGS(trials, target_sequence, τ)
2:      detected ← empty array
3:      for each trial in trials do
4:          target_locations ← find_first_target_symbols(trial.sequence, target_sequence)
5:          if any(x → test_target_location(x, trial, τ), target_locations) then
6:              push!(detected, true)
7:          else
8:              push!(detected, false)
9:          end if
10:     end for
11:     return detected
12: end procedure
```

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× 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

```
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).

negative trials under balanced conditions. Thirdly, I examined whether the amount 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, F1-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 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).

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

F1-Score

The F1-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).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{6}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

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.

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{8}$$

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.

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{9}$$

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{10}$$

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).

$$\text{MCC} = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{(\text{TP} + \text{FP}) * (\text{TP} + \text{FN}) * (\text{TN} + \text{FP}) * (\text{TN} + \text{FN})}} \tag{11}$$

4 Results

In this chapter, I will present and describe the results obtained from the main and side experiments conducted as outlined in the Section 3.2.1.

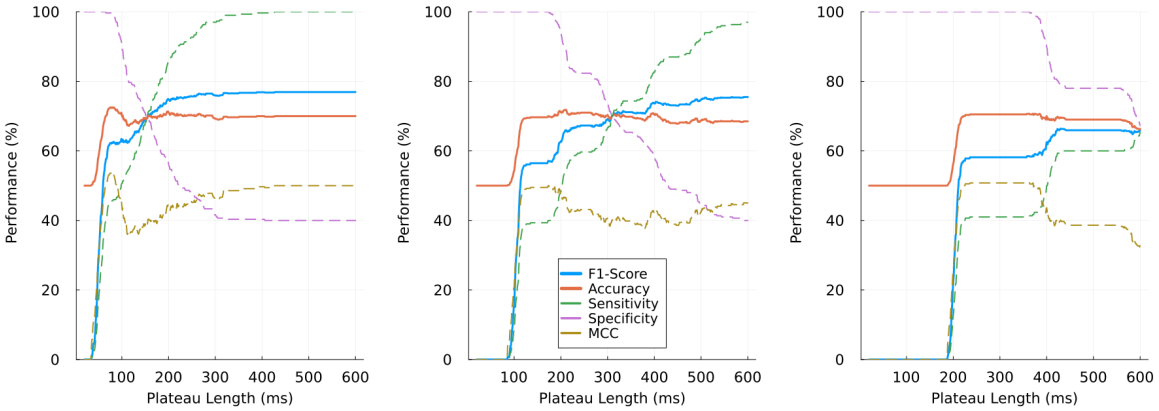
4.1 Vanilla Dendritic Tree

First of all, I will describe the results of the main experiment. Namely, I will start with the results of the vanilla Dendritic Tree E1 which comprises no logic.

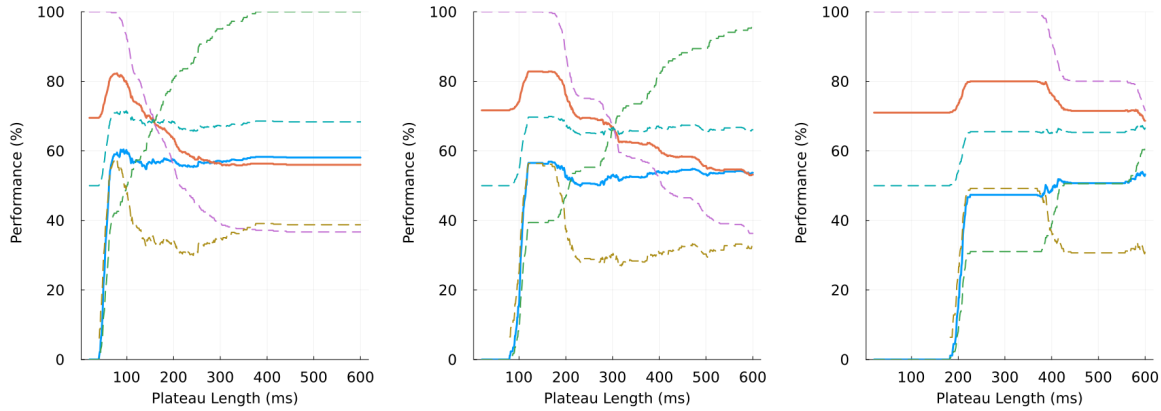
Performance evaluations

The results for σ 10 and 50 are depicted in Figure 6 and 7 respectively. The results for σ 25 can be found in the Appendix, see A.1. I have not included this Figure into the results Section since for σ 10, 25 and 50 there is a similar trend shown whereas the two σ 10 and 50 depict a useful contrast. Moreover, for evaluating the results I realized that the μ range from 50-200 is useful to analyze for a τ range of 20-600, rather than the range 1-400 that I tested. Therefore, in the figures a μ of 50, 100 and 200 is depicted. In the figures I included all three trial generation conditions, balanced, unbalanced and the extra condition.

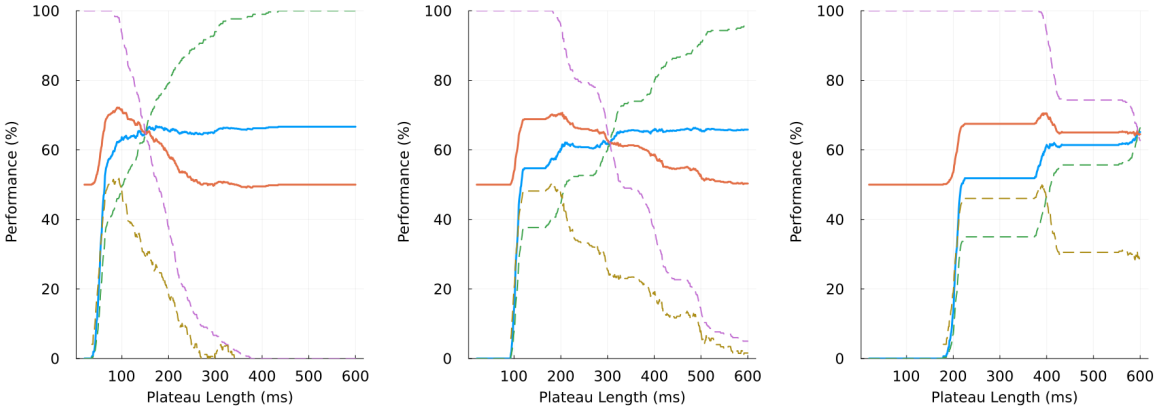
Starting with Figure 6, and σ 10 respectively, it can be observed that the accuracy as well as the F1-Score saturates for all 3 conditions. For the balanced condition the accuracy saturates around 70 %, for the unbalanced around 60% and for the extra condition around 50 %. Nevertheless, the saturation of the F1-Score and accuracy is a clear trend. Moreover, it can also be observed for all three conditions that Sensitivity and Specificity have a crossing point. For increasing μ values the position of the crossing point is shifted towards the right, whereas for a μ of 50 this crossing point is located around a τ value of 150, for a μ of 100 around a τ of 300 and for a μ of 200 around a τ of 600. Overall, specificity initiates at 100%, decreases, and eventually saturates (in the extra condition, it diminishes to zero), while sensitivity begins at 0% and escalates to 100% for larger τ values. This trend suggests that upon reaching a specific τ range, the system accurately identifies 100% TP but fewer TN.



(a) balanced



(b) unbalanced

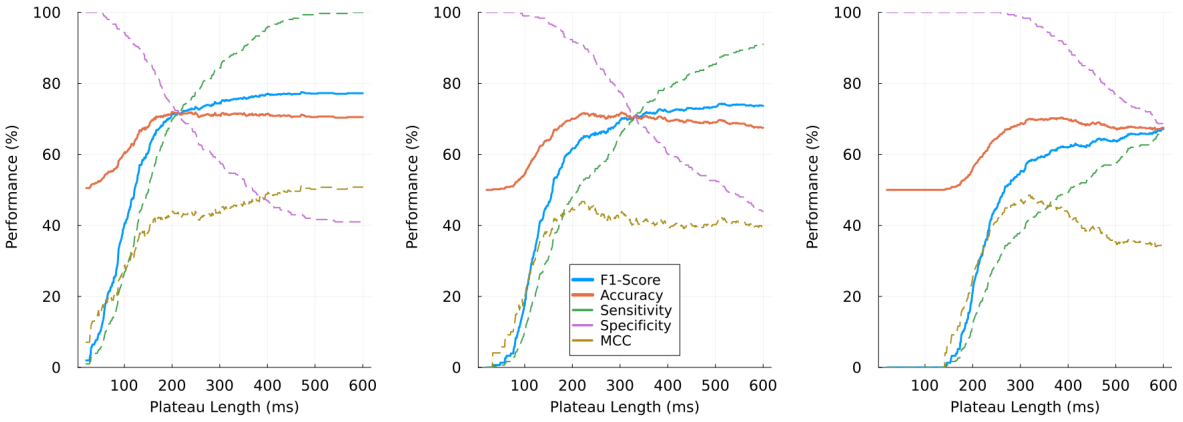


(c) extra condition

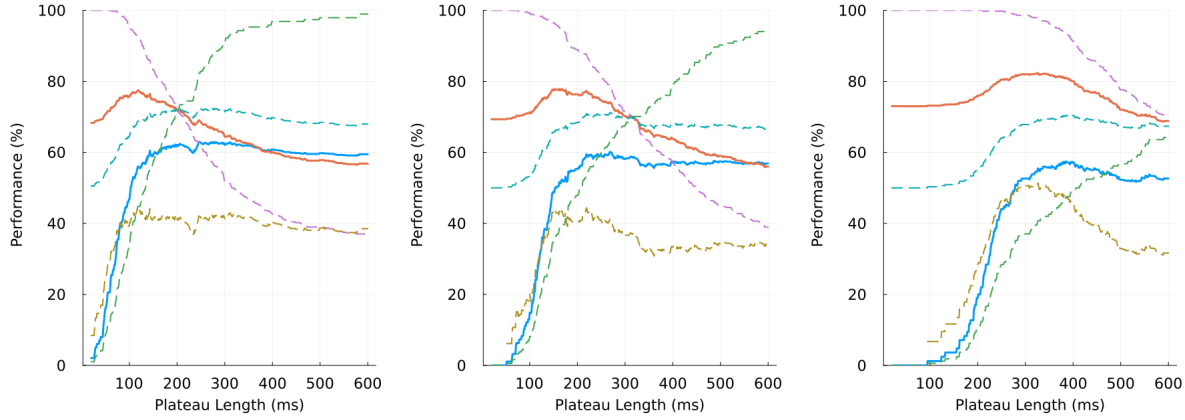
Figure 6: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify Dendritic Tree E1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from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

Notably, in the extra condition, this phenomenon is more pronounced, demonstrating a heightened tendency to achieve 100% TP rate and 0% TN rate. Further, it can also be observed that the accuracy begins to increase after a certain plateau length, which corresponds to the given μ. Across all trial generation conditions, for a μ of 50, the accuracy starts rising from a τ of 50. Similarly, for a μ of 100, accuracy starts to improve from a τ of 100, and for a μ of 200, the increase in accuracy begins at a τ of 200. Moreover, it is important to note that for accuracy, an optimal τ range can be identified in both the unbalanced and extra conditions. Specifically, for a μ of 50, the peak accuracy occurs roughly within a τ range of 50-100. For a μ of 100, the peak range is between a τ of 100-200, and for a μ of 200, the peak range is between a τ of 200-400. Thus, the optimal τ range in terms of accuracy and a σ of 10 consistently extends from the value of the mean up to twice the mean. In relation, the MCC performance also indicates the same τ optimum range. Notably, for the extra condition, the MCC curve does not saturate at a certain point as observed in the balanced and unbalanced conditions. Instead, it decreases from the peak down to 0 (as seen for means of 50 and 100; for a mean of 200, the trend is similar, though the curve extends beyond the plot, suggesting a comparable behavior). For larger means, the behavior of the MCC curve becomes more extended. When directly comparing balanced and unbalanced conditions in terms of MCC values, the decrease is more pronounced in the unbalanced condition. Specifically, while the MCC value decreases from 50% to 40% across all means in the balanced condition, it decreases from approximately 50% to 30% in the unbalanced condition. Lastly, the evaluation of the balanced accuracy for the unbalanced condition reveals a notable observation: the accuracy is consistently higher than the balanced accuracy across all means. This indicates that while the model performs well overall, its performance on the minority class is not as strong.

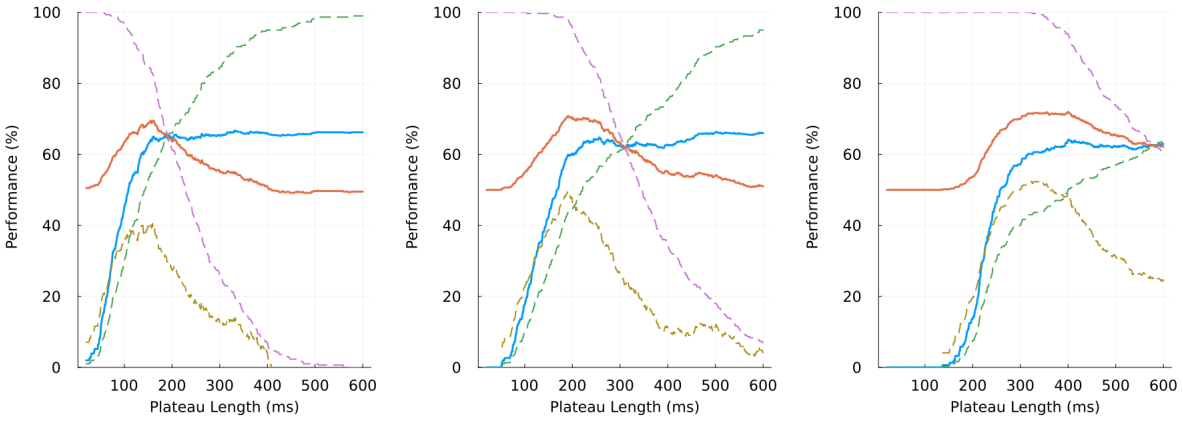
Further, I will proceed with Figure 7 with σ set to 50. In comparison to the findings discussed earlier for a σ of 10, there are notable similarities and differences to observe. Firstly, the crossing point behavior of sensitivity and specificity mirrors that of σ 10, but with a slight rightward shift. The F1-Score demonstrates similar saturation tendencies across all conditions, reaching a plateau within a similar value range. Accuracy also remains higher for the unbalanced condition compared to balanced accuracy. Additionally, MCC values for the extra condition also peak and then decline towards zero. However, there's a distinct difference in the peak range behavior of MCC and accuracy. Unlike the steep incline observed with σ 10, the curves exhibit a smoother, broader ascent and descent, resulting in a wider curve



(a) balanced



(b) unbalanced



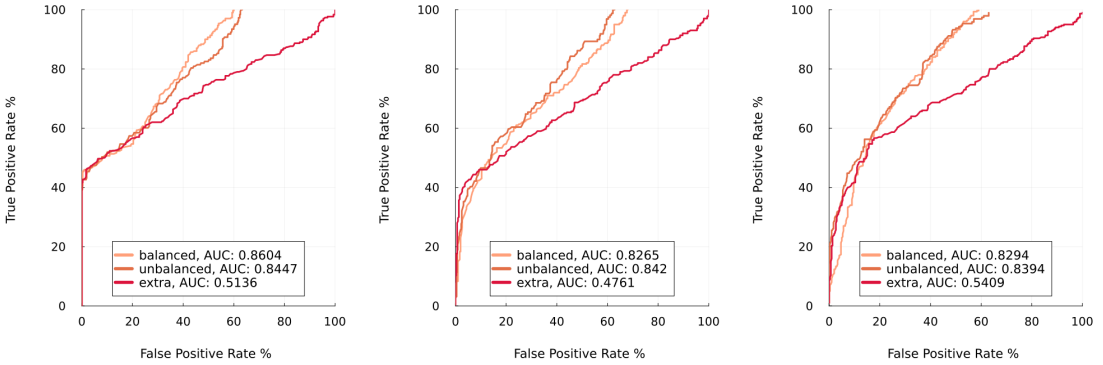
(c) extra condition

Figure 7: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify Dendritic Tree 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 (left), 100 (middle), and 200 (right).

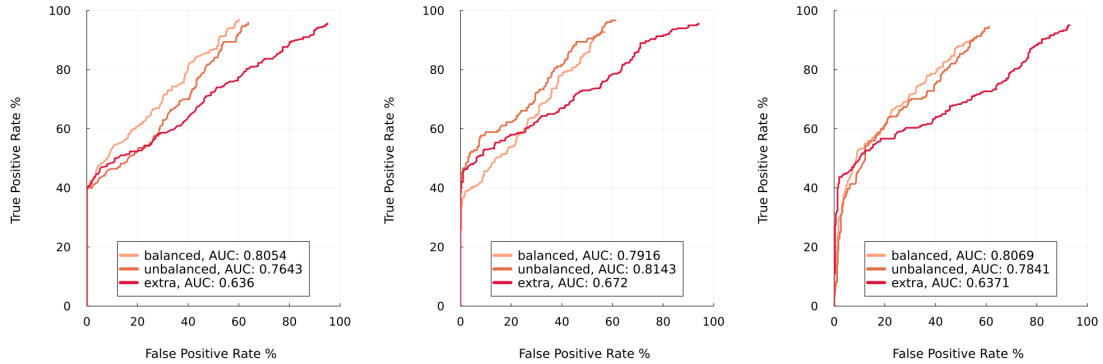
shape. Regarding the optimal τ range for the extra and unbalanced conditions, the indications are weaker due to the smoother curve behavior. Nevertheless, a trend towards an optimum range for accuracy and MCC emerges, roughly between 50+μ and 50+μ*2 for both unbalanced and extra conditions. Since 50 is the standard deviation this interval can be formalized as [σ+μ, σ+μ*2]. I observed this optimal τ range behavior in terms of accuracy and MCC also for a σ of 25, see Appendix Figure A.1. Overall, the influence of changing σ and μ is evident. Specifically, increasing μ shifts the curves to the right and elongates them. Conversely, increasing σ results in a smoothing or widening effect, making the curves flatter, or less steep.

ROC curve and AUC

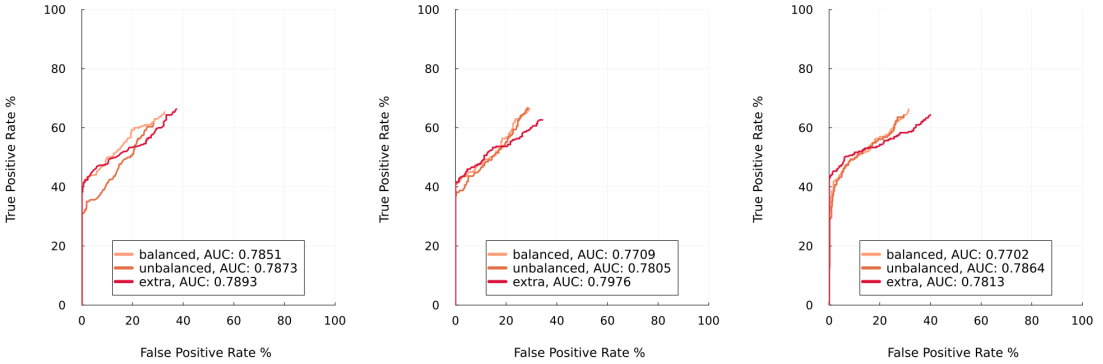
Furthermore, I evaluated the ROC curves and AUC values for the vanilla Dendritic Tree under all trial generation conditions and combinations of σ and μ, as shown in Figure 8. For a μ of 50 and across all three standard deviations and trial generation conditions, it can be observed that all performances reach a point where the true positive rate hits 100%. Additionally, as previously noted, in the extra condition, both the true positive rate and the false positive rate reach 100%. For a μ value of 100, the trend towards high true positive rates is clear, although 100% is not quite reached. Interestingly, at a μ value of 200, the ROC curves appear to be roughly half as wide as those for a μ of 50. The true positive and false positive rates are comparatively lower, peaking at about 65%, while for μ values of 50 and 100, the rates reach up to 90% or 100%. Regarding performance, it is noteworthy that the classifier performed quite well for the balanced condition, with AUC scores ranging from 0.77 to 0.86 across all μ and σ combinations. For the unbalanced condition, the classifier was similarly stable, with AUC scores ranging from 0.76 to 0.84. The extra condition, however, showed more variability, with AUC scores ranging from 0.47 to 0.79, where an AUC below 0.5 indicates performance worse than random guessing. The worst performance (AUC of 0.47) was observed for a σ of 25 and a μ of 50, while the best performance (AUC of 0.79) occurred for a μ of 200 and a σ of 25. When interpreting these results, it is important to remember that the ROC curves are depicted for the entire range of τ values (20-600). Since for the extra condition there is a clear optimum range for τ that is lower for smaller means and standard deviations (as described in Section 4.1), it is not surprising to see performance differences across different means. My focus was on whether the classifier's performance varied with different standard deviations for a given mean. For a μ of 50, the balanced AUC scores are 0.86, 0.82, and 0.82; for the unbalanced condition, the scores are 0.84,



(a) $\mu = 50$



(b) $\mu = 100$



(c) $\mu = 200$

Figure 8: The figure illustrates the ROC curves and AUC scores of the naive classifier in its attempt to classify Dendritic Tree 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 (left), 25 (middle), and 50 (right) and μ values of a) 50, b) 100, and c) 200.

0.84, and 0.839; and for the extra condition, the scores are 0.51, 0.47, and 0.54. This indicates that for a μ of 50, performance remains stable with a maximum variation of 0.07 in AUC scores. For a μ of 100, the balanced condition scores are 0.8, 0.79, and 0.8; the unbalanced condition scores are 0.76, 0.81, and 0.78; and the extra condition scores are 0.63, 0.67, and 0.63, with a maximum difference of 0.06. For a μ of 200, the balanced condition scores are 0.78, 0.77, and 0.77; the unbalanced condition scores are all 0.78; and the extra condition scores are 0.78, 0.79, and 0.78, showing a maximum difference of just 0.01. Therefore, it can be said that for the vanilla branching E1, the σ values of 10, 25, and 50 have only a slight influence on the classifier's performance. The balanced and unbalanced conditions remained stable across all mean and standard deviation combinations, whereas the extra condition showed worse performance for lower means, due to a lower optimum τ range and the presentation of ROC curves for all τ values.

4.2 Complex Dendritic Trees

After describing the results for the vanilla Dendritic Tree that comprises no logic, I will now proceed with the results of the Dendritic Trees that comprise logic, namely D, F and G Dendritic Trees.

Outlier

Throughout the main experiment, I observed that the F1-Score remained saturated at no less than 50% for the balanced and extra trial generation conditions across all combinations of μ and σ for all complex Dendritic Trees. Detailed Figures of Dendritic Trees D1, D2, F1, F2, G1 and G2 for extra, balanced and unbalanced trial generation conditions with a σ of 10, 25, 50 and μ of 50, 100, 200 can be found in the Appendix (see Figures A.2-A.19). Given the consistency of this behavior, I chose not to include all the Figures in the main results Section, focusing instead on outliers and novel insights. Specifically, exceptions were found under the unbalanced condition for the F and G structures. As these exceptions occurred for the F1, F2, G1, and G2 structures, I decided to present the performance of F2 and G2 for σ 25 and μ 100 as representative examples, shown in Figure 9. For Dendritic Tree F2, the F1-Score decreased by up to approximately 5%. The F1-Score exhibited a clear peak and a pattern similar to the MCC curve, which differed from the performance of other Dendritic Trees. However, the crossing point of the sensitivity and specificity curves was consistent, although the sensitivity curve appeared step-like. The accuracy

and specificity curves, which were nearly congruent, started at 100% and decreased to around 80%. The balanced accuracy was lower than the accuracy until a tau of 400, after which the balanced accuracy exceeded the accuracy. This indicates that the performance on the minority class was initially weaker than on the majority class but became stronger beyond a tau of 400. The G2 structure Dendritic Trees showed different behavior under the unbalanced condition. The plot appeared sparse, with accuracy and specificity starting at and minimally decreasing from 100%, while the F1-Score, sensitivity, and MCC curves remained at 0%. The balanced accuracy saturated at 50% and slightly decreased. Upon calculating the evaluation metrics, I encountered a division by zero due to zero TP and zero FP in the confusion matrix. I resolved this by setting the value to 0, as division by zero is undefined. This suggests that for the G2 Dendritic Tree, no positive trials were present under the unbalanced condition.

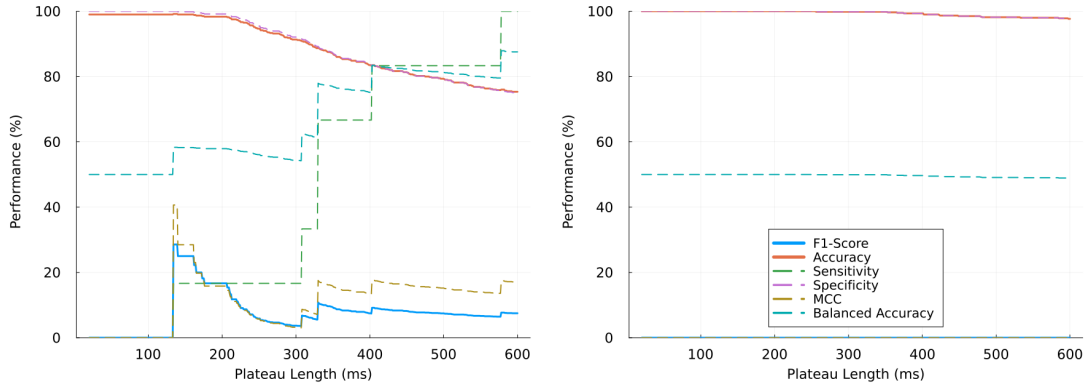


Figure 9: The Figure illustrates the performance evaluations of the naive classifier in detecting Dendritic Tree F2 (left) and G2 (right) for the unbalanced trial generation condition. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14 for F2 and 18 for G2. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F] for F2, and [:A, :B, :C, :D, :E, :F, :G, :H] for G2. Timing parameters followed a Gaussian distribution with a σ of 25 and μ value of 100.

Accuracy

Further I decided to compare the Accuracy performances in between all Dendritic Trees to get an intuition of the differences in performance. Exemplary the accuracy performances for μ 100 and σ 25 can be seen in Figure 10. For the balanced condition, it can be observed that more complex trees, such as F and G, exhibit higher accuracy, saturating around 80% and 90%, respectively. In contrast, structures E and D

saturate between 65% and 55%. Notably, the more complex Dendritic Trees do not show a decrease and a peak in performance as seen with structures E and D. Instead, they display a consistently increasing accuracy. Under the extra condition, all Dendritic Trees exhibit an increase followed by a decrease in accuracy, resulting in an optimum range, unlike the balanced condition. For the unbalanced condition, complex Dendritic Trees start with an accuracy around 100% and saturate between 100% to 90% for the G structures, around 75% for the F2 structure, and 50% for the F1 structure. In contrast, E and D2 Dendritic Tree structures start around 70% and saturate around 60%. Interestingly, Dendritic Tree D1 is a clear outlier, starting at 20% and saturating around 80%. The comparison between D1 and D2 reveals interesting patterns. In the balanced condition, Dendritic Tree D2's accuracy starts at 70%, whereas D1 starts around 50%. Under the extra condition, both start around 50%, but D1 reaches a peak accuracy of 80%, while D2 peaks around 75%. In the balanced condition, D2's accuracy peaks at 85%, whereas D1 peaks around 80%. Therefore, it can be said that D2 performs better than D1 for the balanced condition and for the extra condition D1 performs better than D2. Furthermore, for the more complex Dendritic Trees, a prolonging effect of the optimum range behavior is observed under the extra condition, with the curves appearing wider.

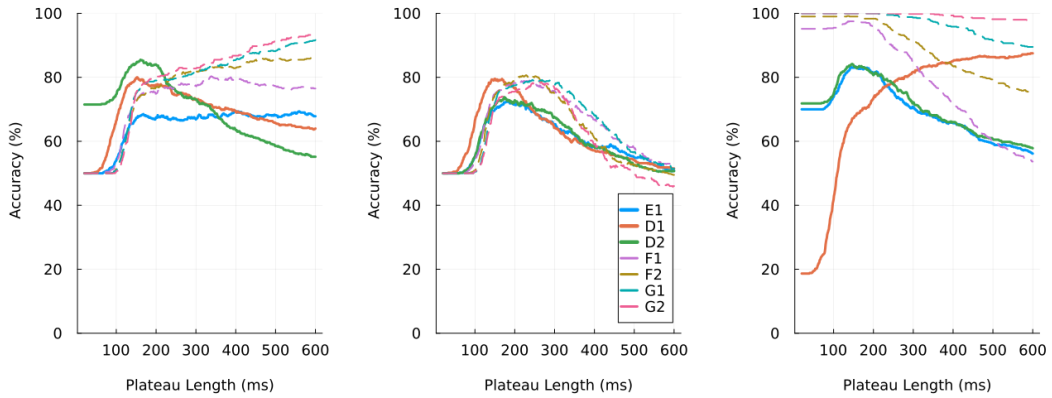


Figure 10: The Figure illustrates the accuracy measurements of the naive classifier in classifying all tested Dendritic Trees across three trial generation conditions: balanced (left), extra (middle), and unbalanced (right). The classification involved a τ ranging from 20 to 600 and a trial count of 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with a σ of 25 and a μ of 100.

The plots for all μ and σ combinations can be seen in the Appendix see (A.20,

A.21, A.22). I have not included them in the main results Section because they depict a similar trend. However, I will discuss the key insights here. Throughout all conditions and σ and μ combinations i could observe the following. Dendritic Tree D1 consistently performs better than D2 for the extra condition, while D2 performs better for the balanced condition. It is important to note that D1 includes only one logical OR, whereas D2 includes only one logical AND. For G1 and G2, as well as F1 and F2 structures, no clear performance difference can be stated. Moreover, the accuracy does not fall below 50% for all Dendritic Trees under both balanced and extra conditions, with the exception of Dendritic Tree G2, which shows a minimal drop in accuracy for three specific μ and σ combinations under the extra condition. D2 is an outlier, starting for the unbalanced condition with an accuracy between 20% and 30%. Additionally, I looked more deeply into the initially observed optimum range discussed in Section 4.1 for E1. First of all, the optimum ranges were only visible for the extra condition. For higher μ and σ values, it becomes more difficult to determine the exact optimum range due to the broader behavior of the curves. However, it was observable that D1 and D2 trees exhibit similar optimum range behavior to E1, although their accuracy peaks differ. Furthermore, this optimum range appears to be prolonged for more complex Dendritic Tree structures F and G, extending the range up to 50 milliseconds. For example, for E1 with a standard deviation of 10, the optimum range is approximately 50-100 ms, whereas for more complex structures, it extends up to 150 ms. I also observed that for the balanced condition that complex Dendritic Trees F and G consistently achieve higher accuracy than simpler Dendritic Trees E and D and that this effect diminishes as the mean increases.

MCC

Lastly, I compared the MCC curves for all Dendritic Trees. An example can be seen in Figure 11, with σ of 25 and μ of 100. For the balanced condition, complex structures (F and G) showed an increase and saturation around 60-85%, while simpler structures (E and D) saturated around 40%. Among the simpler structures, D structures had a higher peak than E1, with D1 reaching a higher peak than D2. For the extra condition, all Dendritic Trees exhibited an initial increase in MCC followed by a decrease to zero. The complex structures comprised wider and higher peaks compared to the simpler ones. Notably, Dendritic Tree D1 achieved a greater peak than E1 and D2. For the unbalanced condition, MCC performances of the G structures were not depicted because their values were zero, as no positive trials were integrated (as

discussed in Section 4.2). The behavior of E and D structures was similar to the balanced condition. However, F structures displayed distinct differences: F1 had a sharp peak at 70%, and F2 at 40%, with both decreasing to approximately 20%.

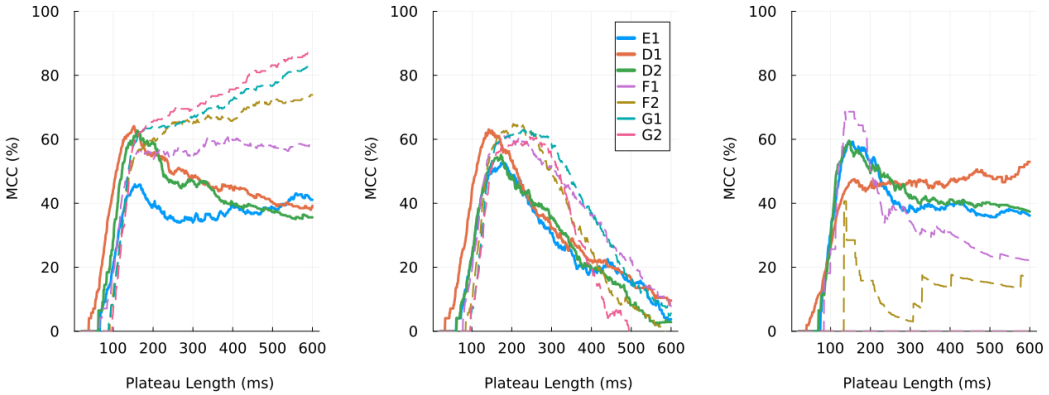


Figure 11: The Figure illustrates the MCC measurements of the naive classifier in classifying all tested Dendritic Trees across three trial generation conditions: balanced (left), extra (middle), and unbalanced (right). The classification involved a τ ranging from 20 to 600 and a trial count of 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with a σ of 25 and a μ of 100.

In the Appendix I also included the plots of the MCC scores for all Dendritic Trees for all σ and μ combinations, see Figures A.24 A.23 A.25. These plots were not included in the main results Section because they mostly followed a trend, but key findings are outlined here. For the balanced condition it can be observed that from lower to higher σ values, the wide performance differences between complex and simple structures diminish, similar to trends observed in the accuracy curves. For the extra condition, MCC scores exhibit a clear peak and approach 0 afterwards. In the unbalanced plots, the G structures were not visible, with two exceptions where MCC score of G1 are integrated. It was challenging to evaluate the unbalanced MCC scores due to the lack of observable trends. Overall, it can be said that complex structures generally outperform simpler ones in terms of MCC. For the balanced condition, complex Dendritic Trees saturate around 70-90%, whereas simpler ones saturate between 30-40%, indicating a clear difference. In the extra condition, all peaks for the complex Dendritic Trees are at least above 50%, and the MCC values of the simpler Dendritic Trees are at least above 40%, which still outperforms the simpler structures. Moreover, I observed that D1 consistently outperforms D2 in both

balanced and extra conditions. MCC can be interpreted as follows: -100% indicates inverted performance, 0% means the system performs no better than a random guess, and 100% indicates perfect performance in evaluating all values of the confusion matrix. Consequently, all Dendritic Trees perform better than a random guess as long as the distribution samples are at least as large as τ for the balanced condition, and for the extra condition additionally within the range of the peak indication.

4.3 Side experiments

Lastly, I will provide the key insights gained from the side experiments.

Trial count

I tested the performance of the classifier for balanced and extra condition for the vanilla branching structure E1 with a trial count of 300, 600 and 900. I have not noticed a significant difference in the performances. The plot can be seen in A.26.

Negative Bias

I introduced for the vanilla branching structure E1 a negative bias of 90% and 99% for balanced and extra condition. Notably, the resulting negative bias plots showed similarities to the unbalanced outlier plots of Dendritic Trees F1 and F2. The F1-score decreases and a step like sensitivity could be observed as well as a steep decrease of the accuracy which is even strengthen for the extra condition. Plots can be found in the Appendix, see Figure A.28.

Alphabet

I tested the performance of the naive classifier for the vanilla branching structure E1 with α :A-:D; :A-:F and :A-:I for balanced and extra condition. The results reveal that accuracy, F1-score and MCC score increase for increasing length of the alphabet, see Appendix Figure A.27.

Uniform distribution experiment

I evaluated the performance of the naive classifier with a τ of 200 for the vanilla branching structure E1 for both wider and narrower parameters a and b of the uniform distribution, see Appendix Figure A.29. In the balanced condition, I observed a clear trend: as the distribution parameters a and b increase (ranging from 10-50, 10-100,

10-200, to 10-300), the sensitivity of the classifier decreases while specificity increases. This implies that the classifier is more effective at correctly identifying true trials with narrower distribution samples, and better at correctly identifying false trials with wider distributions. A nearly balanced outcome of sensitivity and specificity was noted for the parameter range of 10-100. For the extra condition, the performance varied from the balanced condition, especially for narrower distributions. Specifically, the specificity for narrower distributions (10-50 and 10-100) was significantly lower in the extra condition. For instance, the specificity for the 10-50 parameter range was approximately 60% in the balanced condition but dropped to around 5% in the extra condition. This indicates a poorer performance in classifying negative trials for narrower distributions in the extra condition. However, for wider distributions, the classifier's performance was similar to that in the balanced condition. For both trial generation conditions, when the distribution parameter range exceeded the threshold $\tau = 200$ (specifically in the range of 200-300), the specificity reached 100% and sensitivity 0 %. This occurred because all timings exceeded the threshold, causing every trial to be classified as false, as τ serves as a cutoff point above which no timings are classified as true.

5 Discussion

In this chapter, I will interpret and discuss the results presented in the previous chapter, address the limitations of this research, and provide an outlook.

5.1 Interpret results

Firstly, I discovered that varying the parameters σ (10, 25, 50) and μ (50, 100, 200) in the Gaussian distribution for the input timings influences the performance of the naive classifier. Exemplary i have shown this effect in detail for E1, but I have observed this for all Dendritic Trees. Increasing the parameter μ shifts and elongates the curves to the right. I would explain this rightward shift with the fact that the classifier can only classify a trial as true when the threshold τ exceeds the timing differences of the inputs as explained in Section 3.1.5. Thus, for $\tau < \mu$, the probability is high that all trials are classified as false since for the Gaussian distribution, sampled values are most likely to be close to the mean. As μ and the corresponding input timings increase, the accuracy and sensitivity curves rise only when τ is at least as large as the input timings, which might cause the rightward shift. Conversely, increasing σ results in a smoothing or widening effect, making the curves flatter or less steep. I would explain this effect by the fact that a higher standard deviation allows values to be sampled from a wider range. Consequently, instead of having a distinct peak, as seen with smaller standard deviations where the range is narrow, higher standard deviations result in smoother and flatter curves. Moreover, upon examining the ROC curves of Dendritic Tree E1 as an example, I observed that varying the σ values of 10, 25, and 50 had only a minimal impact on the classifier's performance. The performance of both the balanced and unbalanced conditions remained consistent across all combinations of mean and standard deviation. However, in the case of the extra condition, there was a deterioration in performance for lower mean values, attributable to a narrower optimal τ range and the presentation of ROC curves for all τ values. When keeping the mean constant and varying the standard deviation, a similar level of robustness was observed in performance, even for the extra condition.

These results suggest that the classifier maintains robustness across different mean and standard deviation configurations for the balanced and unbalanced conditions of E1. However, in the case of the extra condition, while an optimal threshold range is evident, robustness in performance can still be observed across variations in standard deviation under the same mean.

Secondly, I examined the experimental results not only for the vanilla Dendritic Tree E1 but also for the other complex Dendritic Trees of types D, F, and G. A striking observation was that the F1-Score remained saturated at no less than 50% under both balanced and extra trial generation conditions across all combinations of μ and σ for all complex Dendritic Trees. I would make sense of this observation by considering that the classifier reaches a TPR of 100% at a certain point as τ increases. Since the F1-Score is calculated based on sensitivity, this explains why the saturation persists with increasing τ (e.g., no decrease could be observed). However, a decrease in the F1-Score was observed for the unbalanced outlier examples, as discussed in Section 4.2. The results for the outliers can be attributed to either the presence of only a few positive trials in the set for the F structure or the complete absence of positive trials in the set for the G structure. Since the outliers were observed under the unbalanced trial generation condition, it can be inferred that the F and G structures became more complex, making the occurrence of positive instances in the trials highly unlikely by chance. Connecting these results with the side experiment on negative bias, the behavior of the curves for the F structure was observed to be similar, confirming my interpretation. Several conclusions can be drawn from these insights. Based on the saturation of the F1-Score, it is more insightful to examine the MCC score when evaluating performance, as it considers all elements of the confusion matrix equally. This is particularly relevant if one is also interested in the classifier's performance concerning FP. However, if the focus is solely on the classifier's performance for TP, then choosing a large τ will lead to a 100% TP rate. For a more comprehensive evaluation that includes the classification of false trials, analyzing the MCC score and evaluating the optimal range of the classifier for both TP and FP is advisable, which I will discuss in the following.

First of all, by comparing the accuracy scores of all Dendritic Trees with each other, I revealed that the optimum range could only be observed for the extra condition. Specifically, for Dendritic Trees D1 and D2, the initially observed optimum range of roughly $[\text{mean}+\sigma, \text{mean}+2\sigma]$ for E1 was also applicable. For more complex Dendritic Trees like types F and G, I observed a prolonged optimum range of approximately 50 ms. I hypothesize that this might be related to the requirement that for a sequence to

match the extra condition, there must be at least one distractor element in between the occuring target sequence. For more complex Dendritic Trees, I used longer sequences since the possible target combinations are also longer. However, this extension in length also increases the possibility of having several elements in between as distractors rather than just one or two. An extreme case would be a target combination for G1, for instance, [: A, : B, : E, : D, : C, : F, : G] and a possible extra condition sequence [: **A**, : C, : C, : D, : H, : E, : A, **:B**, : A, **:E**, : B, **:D**, : B, **:C**, : D, **:F**, : H, : **G**], where for one element, there are six elements in between. Although the probability of this extreme case is low, it demonstrates that for the complex Dendritic Trees, the likelihood of having more distractor elements in between is higher than for the simpler Dendritic Trees because I did not restrict the number of distractors. I would explain the prolonged optimum range of complex Dendritic Trees compared to simpler Dendritic Trees with this phenomenon since it implies that the classification of false sequences would also work for larger τ values because the number of elements can be distinctly higher than one, probably directly leading to a classification as false since τ cannot be overcome. However, it might be interesting to test the experiment with a threshold for distractor elements, such as a maximum of two, to confirm this hypothesis. Even with this prolongation of 50 ms for the complex trees, it can be said that the approximate range of $[\text{mean} + \sigma, \text{mean} + 2\sigma] \pm 50$ ms can be determined for the Gaussian distribution as an optimum range in terms of accuracy and TP / FP balance.

Concerning the balanced trial generation condition, I observed that there is a bias for increasing complexity of Dendritic Trees. Specifically, since the balanced condition randomly assigns half true and half false sequences, the probability is higher for more complex Dendritic Trees that the target sequence is not in the generated false sequence. Based on the classifier's procedure, if the sequence is not in the correct order, it is directly classified as false. This might explain why the complex Dendritic Trees perform significantly better than the simpler Dendritic Trees. I also observed that for the extra condition, this effect of better performance is smaller, as explained above. Furthermore, for more complex Dendritic Trees, the probability is lower that an element inserted between the target elements in a sequence is part of the target of the next consecutive target element. For example, for a target E1 sequence [: A, : B, : C] and a generated sequence [: B, : C, : D, : **A, :A, :A, :B, :C**, : C, : D], there are fewer elements in the alphabet compared to more complex Dendritic Trees, and the probability is higher that after an ":A", another ":A" occurs. In this example, an edge case is shown where the first ":A" triggers a plateau potential, which might

not be long enough to capture the next ":B" and ":C" elements of the sequence, even though the target sequence is present and should be detected as true. This observation can be connected to the side experiment of varying α for E1, where the same described effect of increasing performance was observed for longer α similarly as for more complex trees. I also observed the effect that the performance difference diminishes for increasing μ values. I hypothesize that this might be due to the right shift induced by the mean, as discussed above. I assume that a further increase in performance would probably be visible for higher τ values that were not included in this experiment. To address the overall observation and the question of whether complex Dendritic Trees perform better, I have shown that several effects might have influenced the performance of the more complex Dendritic Trees. Based on the discussions above, it is not possible to state definitively that they perform better in the context of these experiments. To answer this conclusively, the experiment needs to be redone, considering and addressing the factors I have described.

By contrast, what can be said and what I observed was that in terms of robustness, all Dendritic Trees have demonstrated reliable time-invariant sequence detection. Using ROC curves and AUC scores I have shown that throughout varying standard deviations and means the robustness remained for the Dendritic Tree E1, as well as the Accuracy and MCC scores for all tested Dendritic Trees. My experiments have confirmed that plateau potentials and temporal overlaps, as part of dendritic sequence processing, confer robustness to time-invariant sequence detection, thereby confirming my initial hypothesis. I used named evaluation metrics to assess the robustness of the naive classifier across different mean and standard deviation combinations, leading to my conclusion. I also discovered that the system operates effectively with an optimal τ, achieving 70-80% accuracy for the extra condition. Additionally, I found that factors such as α, negative bias, and the dispersion of the distribution for input timings influence the system's robustness. These results adequately answer my research question. Concerning the initially theorized 'optimal' range for τ, spanning from 200 to 600 milliseconds, my findings indicate a correlation between robustness and an optimal τ range, which is intricately linked to input timings derived from a specific distribution. Through experimentation with a Gaussian distribution, I observed an optimal range of $[\mu + \sigma, \mu + 2\sigma] \pm 50$.

5.2 Limitations

My thesis meets several limitations. First of all, the chosen distribution for the input timings, the Gaussian distribution, is an unrealistic distribution in terms of dendritic sequence processing. Additionally, the truncation of negative timings restricts the distribution. It would be interesting to test the setup with a more realistic distribution, such as the Gamma distribution, which has a more natural dispersion behavior in the context of biological time intervals (and only positive samples possible). Furthermore, it might be interesting to see whether the optimum τ ranges behave similarly or differently for a Gamma distribution, given that I found the optimal τ range to be closely connected to the distribution of input timings.

Moreover, the given model and classifier are naive in terms of being a simplified model rather than a more realistic biological model. No accurate biological conclusions can be drawn from these experiments. Additionally, certain edge cases are not handled yet, as discussed in Section 3. These unhandled edge cases might influence the performance and need to be addressed. Specifically, the edge case where a target sequence, here : A, : B, : C, appears twice in the sequence is notable. For example, in a sequence [**:A**, : C, : D, **:B**, : D, **:C**, **:A, :B, :C**, : D], the naive classifier would probably not detect the sequence as true since it only considers the first occurrences of the target symbols, which have a few distractor elements in between. The consecutive series of : A, : B, : C in the example sequence would not be considered at all, which is crucial to note. The influence of these edge cases is that the classifier is prone to several consecutive identical symbols that are part of the target symbols. Additionally, the classifier only checking for the first occurrence may lead to more false classifications. Regarding several branching combinations, the current coding approach classifies the sequence as true if any combination is true. This could lead to a problem where a combination that is actually in the sequence but falsely classified because τ is not surpassing the timing difference of timing inputs, and then another combination that is not directly consecutive is classified as true. The result is that the classifier might make correct predictions based on false reasons which would falsify the performance. Because it has detected a sequence correctly in these cases but based on a mistake. One might integrate in the context of several branching combinations whether the classifier detects the correct combination rather than any to more accurately asses the classifier's performance.

5.3 Outlook

However, the tendency could be affirmed in this work, as discussed by Bouhadjar et al., 2022; Burger et al., 2023; Leugering et al., 2023, that sequence detection based on plateau potentials and temporal overlaps confers a robust time-invariant sequence detection. Whereas they employed more biophysical and realistic models, I have shown this for a simplified model. Specifically, I found that the classification remains robust for more complex Dendritic Trees. It might be interesting to verify this for more biologically realistic models, as more intricate branching patterns may allow individual neurons to detect more complex sequential patterns and provide more capacities (Leugering et al., 2023; Moore et al., 2022; Quaresima et al., 2023). It is worthy of discussion whether several OR constructs might fulfill a useful purpose due to their nature of increasing possible combinations. I hypothesize that for more complex Dendritic Trees, rather than using multiple OR constructs, employing AND constructs with single OR constructs or simpler constructs with either an AND or OR operator might be more useful. My findings highlight the relevance of further investigating the role of complex arborization in neural computational models and their implications for dendritic memory implementations.

Interestingly, the findings by Burger et al., 2023 were also derived from input timings following a Gaussian distribution, as in my experiments. It would be worthwhile to test such experiments with biophysical models and a gamma distribution because the gamma distribution more accurately represents the skewed and non-negative nature of neural input timings. This approach provides deeper insights into the variability and burstiness of synaptic inputs, which are crucial for understanding the temporal dynamics of neuronal processing. It would be interesting to see whether the results remain consistent or change under these conditions.

Additionally, the generation of trials, as discussed intensively above, raises questions about what set of sequences and respective ratios of true and false trials would be realistic. In the context of dendritic sequence detection with plateau potentials, accurately simulating these sequences is crucial for understanding neuronal behavior. This becomes particularly relevant when considering processes like sleep replay, where the brain reactivates sequences of neural activity experienced during wakefulness. Moreover, sequence learning, prediction, and replay have been proposed to constitute the universal computations performed by the neocortex (Bouhadjar et al., 2022), where Bouhadjar et al., 2022 also integrated sequence replay in their model, meaning that after learning, the model can replay sequences in response to a cue signal. Leugering

et al., 2023 discusses plateau computation as allowing single neurons to solve detection across varying movement speeds and during replay, further underscoring the relevance of plateau computation when modeling neocortex behavior. This leads back to the question: "What is a realistic distribution of true and false trials in simulated neural models for dendritic sequence detection compared to real neural activity?". Studies have indicated that the frequency and content of replay events, including both forward and reverse sequences, can dynamically change over time with experience, highlighting the flexible and adaptive nature of replay in memory processes (Pfeiffer, 2020; van der Meer et al., 2020). Specifically, it is known that the ratio is influenced by experiences like avoidance behavior, leading to more negative trials, or more positive trials to anticipate a future step. Generally it can be inferred that replay is characterized by flexibility, but it remains unclear how specific paths are selected for expression (Pfeiffer, 2020; van der Meer et al., 2020). Therefore, the question remains open. Further research incorporating more complex neural dynamics and experimental validations is necessary to fully assess the realism of trial distributions in simulated models compared to real neural activity.

Another aspect highlighted in the literature is learning. For passive delay-based models, there exists work showing 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 maximizing it using deep learning techniques (Hammouamri et al., 2023). In the context of plateau computation it might be interesting to equivalently learn and optimize the optimum τ range for active dendritic models to provide more efficient sequence detection systems. In addition to learning an optimum τ range, it is a generally discussed topic of how to integrate learning for dendritic behavior since it is unclear how dendrites can learn to detect specific sequences only based on local plasticity mechanisms (Leugering et al., 2023). In the model by Bouhadjar et al., 2022, they devised the HTM model by replacing the original discrete-time neuronal and synaptic dynamics with continuous-time models with biologically interpretable parameters such as membrane and synaptic time constants and synaptic weights. They further substituted the original plasticity algorithm with a more biologically plausible mechanism, relying on a form of Hebbian structural plasticity, homeostatic control, and sparse random connectivity. Moreover, their model implements a winner-take-all dynamics based on lateral inhibition that is compatible with the continuous-time neuron and synapse models. They showed that the revised HTM model supports successful learning and processing of high-order sequences with a performance similar to that of the original HTM model. They

provide a novel step towards successful learning of dendrites; however, more research on dendrites and more biorealistic models are needed to approach neuro-realistic learning of dendrites in computational models.

6 Conclusion

Through empirical experimentation and computational modeling this study has examined the robustness of Dendritic Trees with memory in classifying sequences under timing variations. My experiments have validated that plateau potentials and temporal overlaps, crucial to dendritic sequence processing, confer robustness to time-invariant sequence detection, affirming my initial hypothesis. The assessment of named evaluation metrics across different mean and standard deviation combinations led me to these conclusion. I revealed that the system operates for all Dendritic Trees effectively with an optimal τ, achieving an accuracy of 70-80% for the extra condition. Additionally, factors such as α, negative bias, and the dispersion of input timings have been identified to influence the system's robustness.

In acknowledging the limitations of this study, it's essential to recognize the use of a simplified model, which may not fully capture the complexities of biological dendrites. Future research should strive to incorporate more biorealistic models to validate these findings. Moreover, the choice of distribution for input timings, particularly the Gaussian distribution, may not accurately represent biological phenomena. Exploring alternative distributions, such as the Gamma distribution, could yield deeper insights into synaptic input variability.

Looking forward, directions for further exploration include investigating the role of complex arborization in neural computational models and its implications for dendritic memory implementations. Additionally, exploring learning mechanisms for dendritic behavior and integrating them into computational models remains an open area of research. Moreover, further studying the distribution of true and false trials in neural activity during memory processes and replay events could yield valuable insights for simulated models capable of adapting to more realistic ratios and model environments.

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

Pfeiffer, B. E. (2020). The content of hippocampal "replay" [Epub 2018 Jan 10]. *Hippocampus*, *30*(1), 6–18. https://doi.org/10.1002/hipo.22824

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

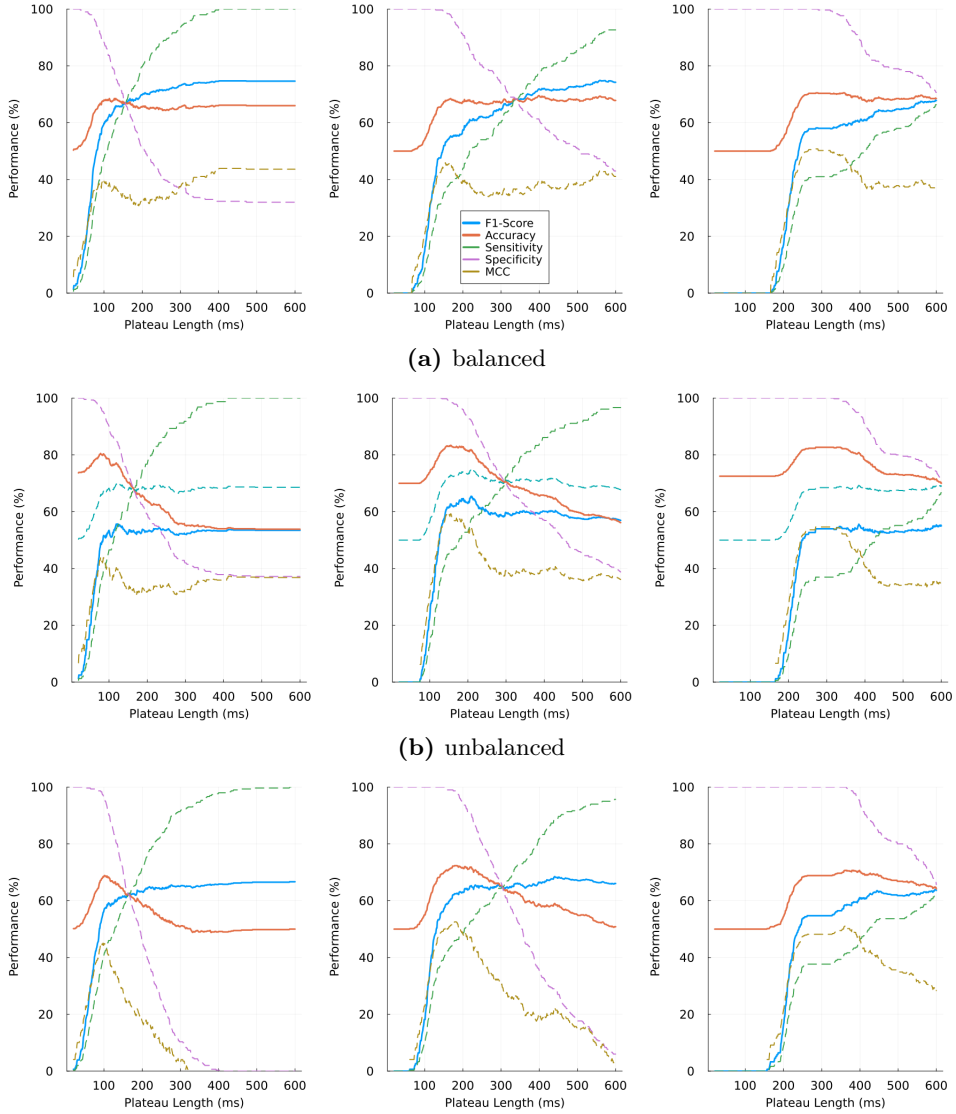
Stuart, G., Spruston, N., & Häusser, M. (2016). *Dendrites*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198745273.001.0001

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

van der Meer, M. A. A., Kemere, C., & Diba, K. (2020). Progress and issues in second-order analysis of hippocampal replay. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *375*(1799), 20190238. https://doi.org/10.1098/rstb.2019.0238

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

A Additional Figures

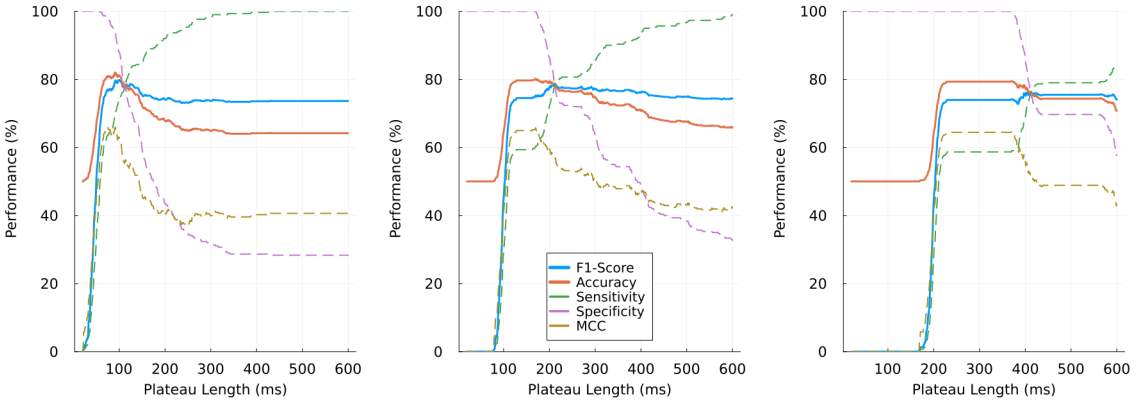


(a) balanced

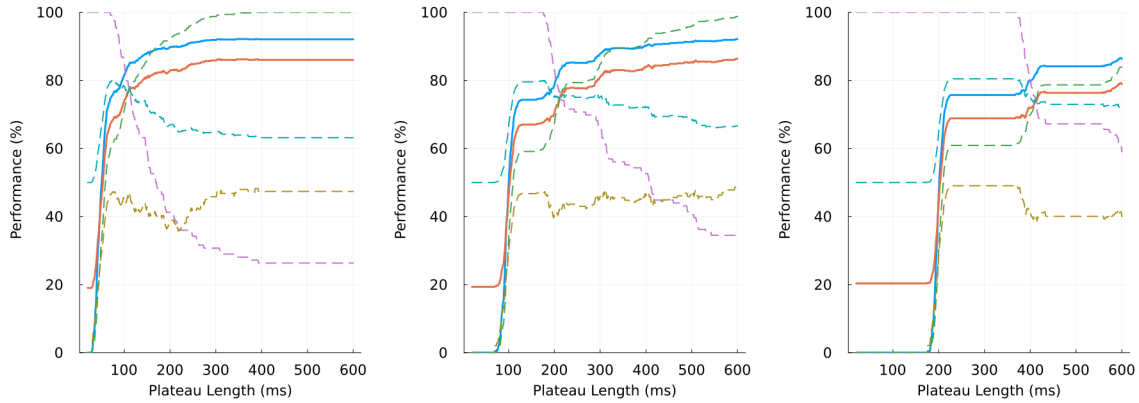
(b) unbalanced

(c) extra

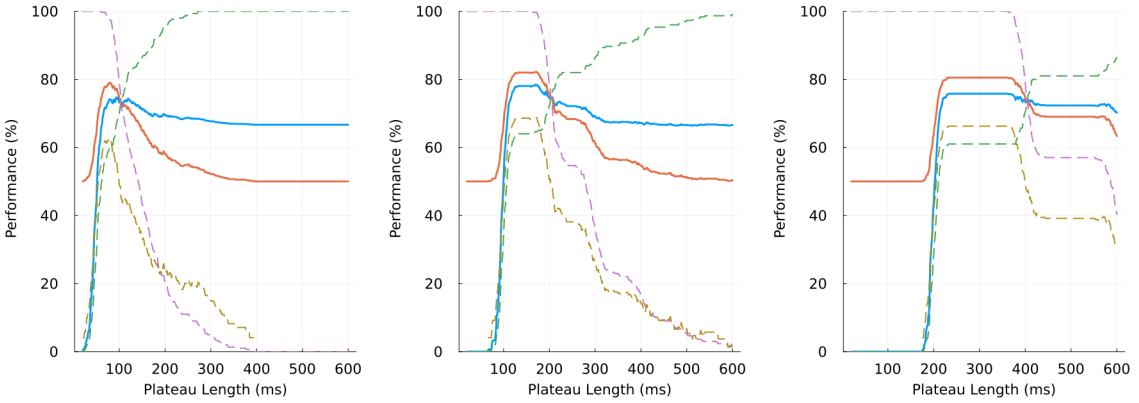
Figure A.1: The figure illustrates the performance evaluations of the naive classifier in its attempt to classify the Dendritic Tree E1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated 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 (left), 100 (middle), and 200 (right).



(a) balanced

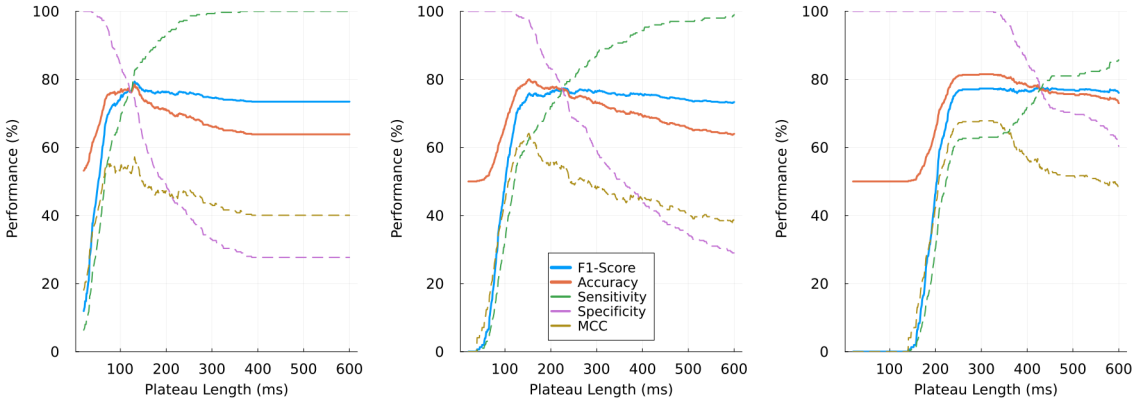


(b) unbalanced

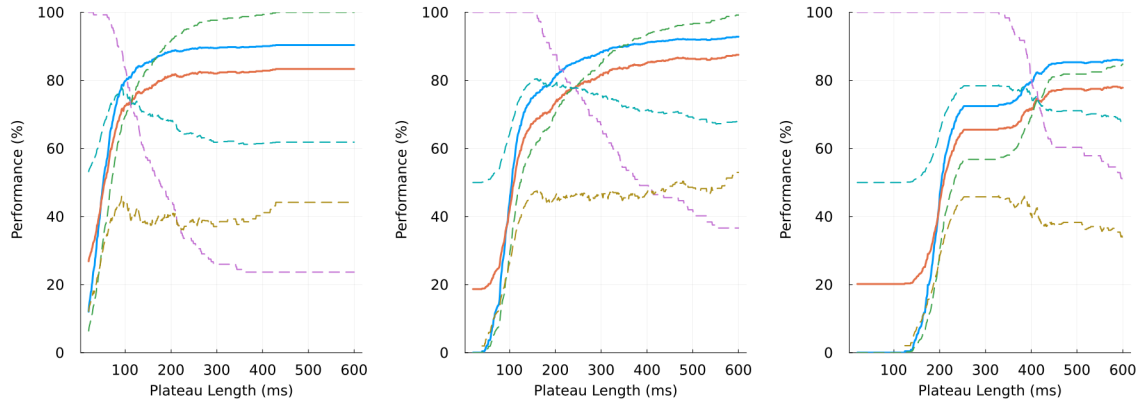


(c) extra condition

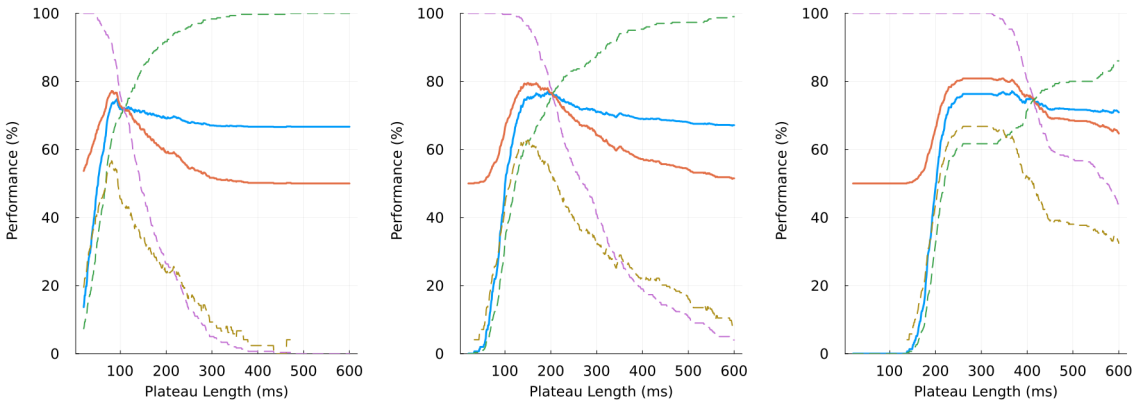
Figure A.2: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

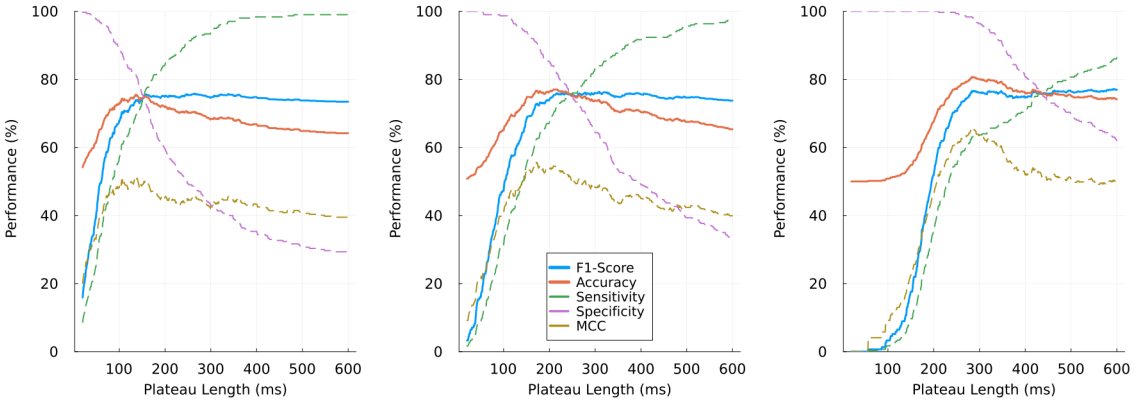


(b) unbalanced

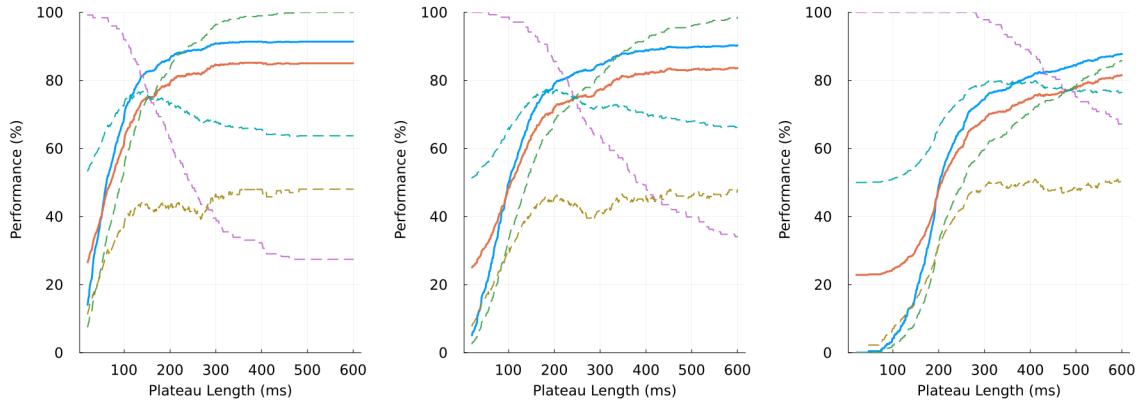


(c) extra condition

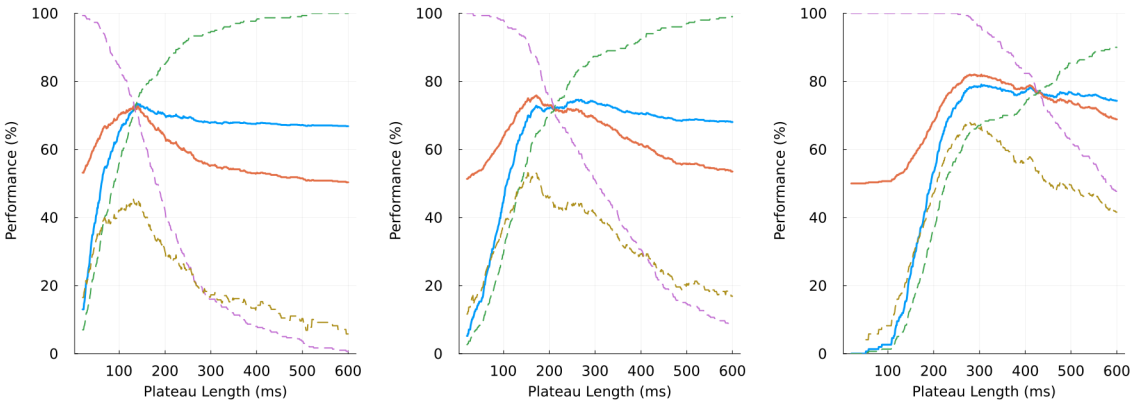
Figure A.3: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

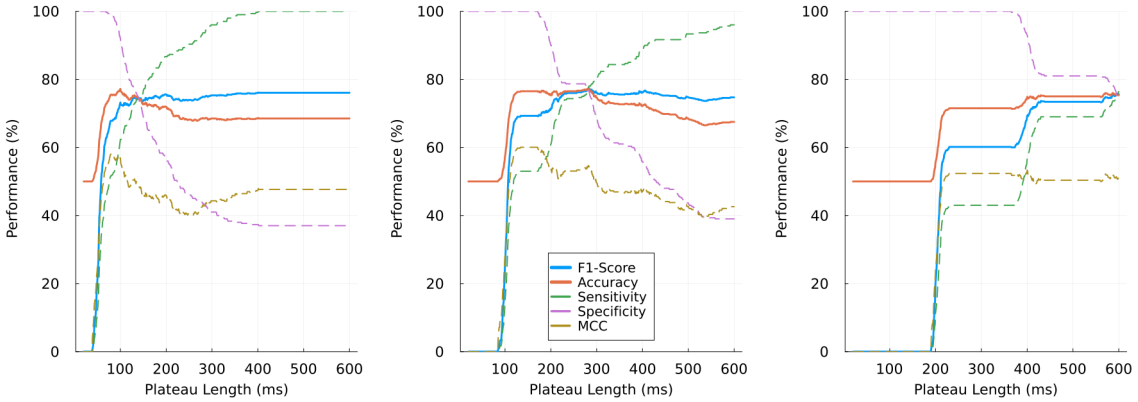


(b) unbalanced

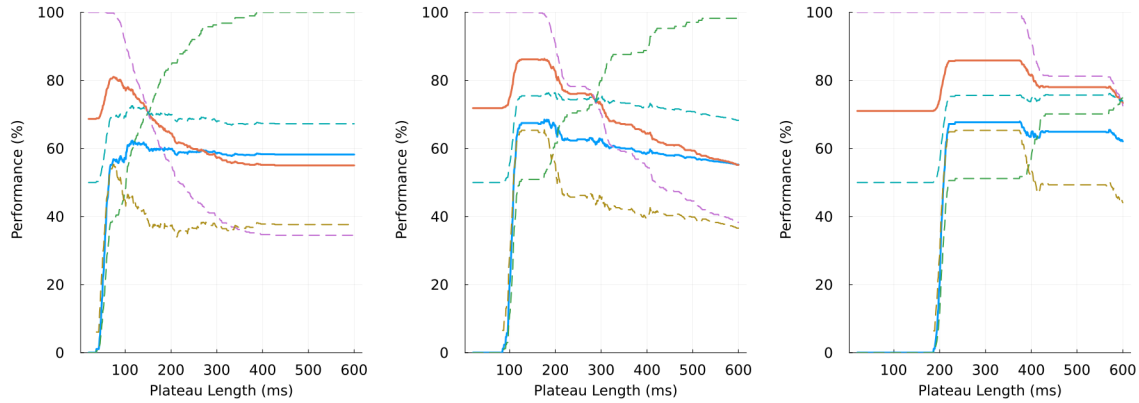


(c) extra condition

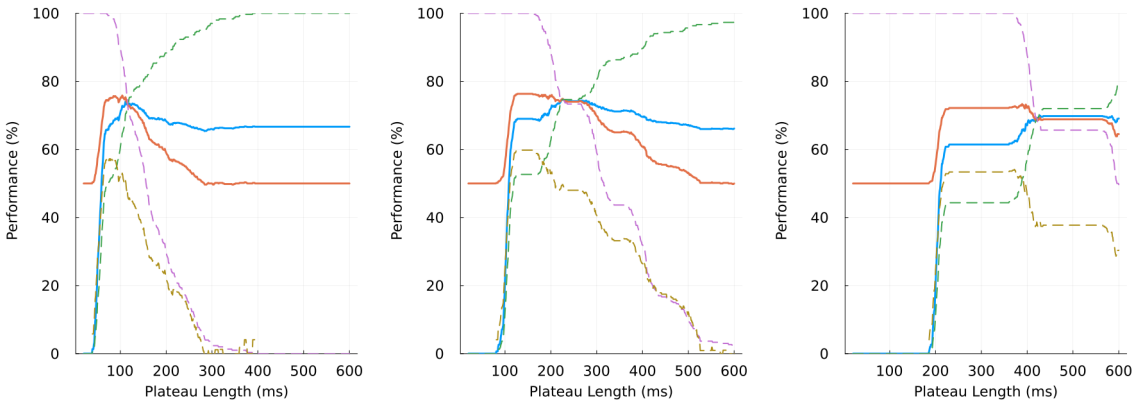
Figure A.4: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

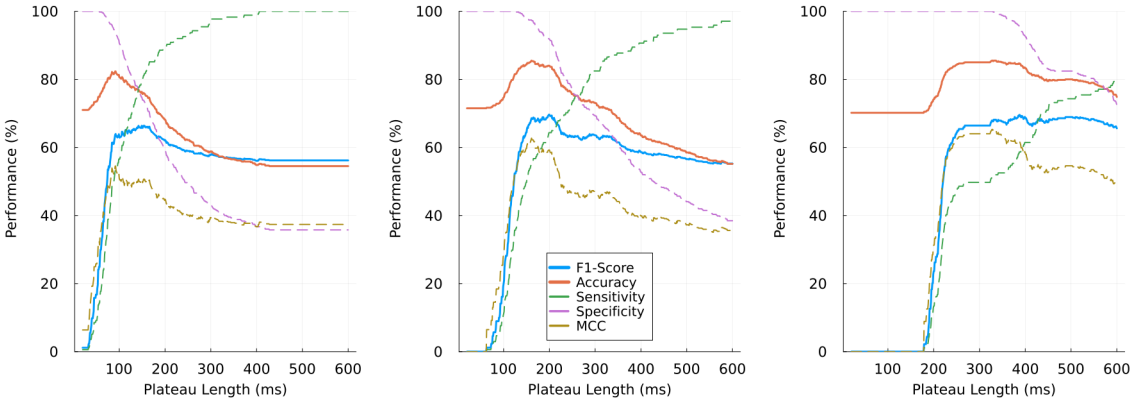


(b) unbalanced

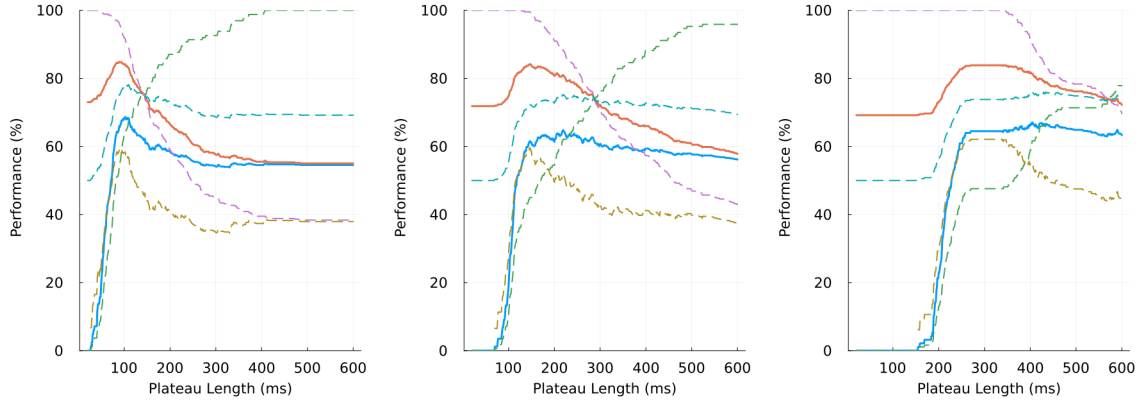


(c) extra condition

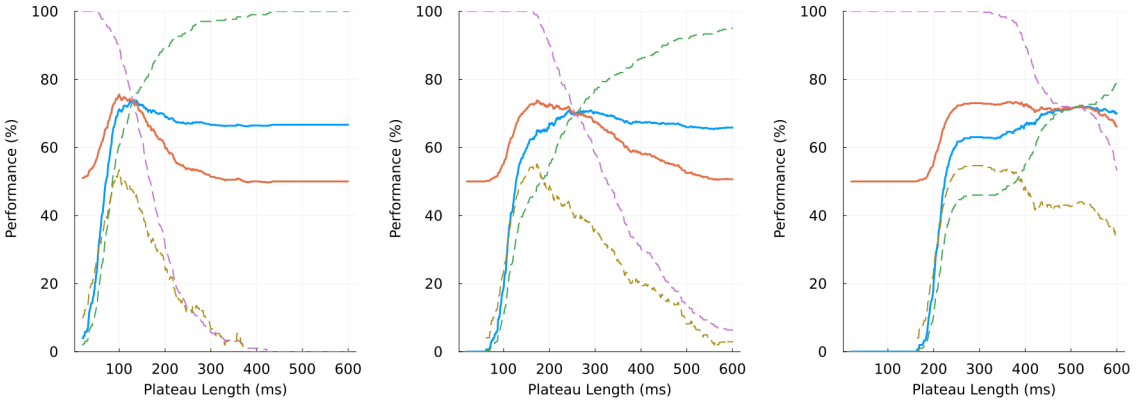
Figure A.5: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

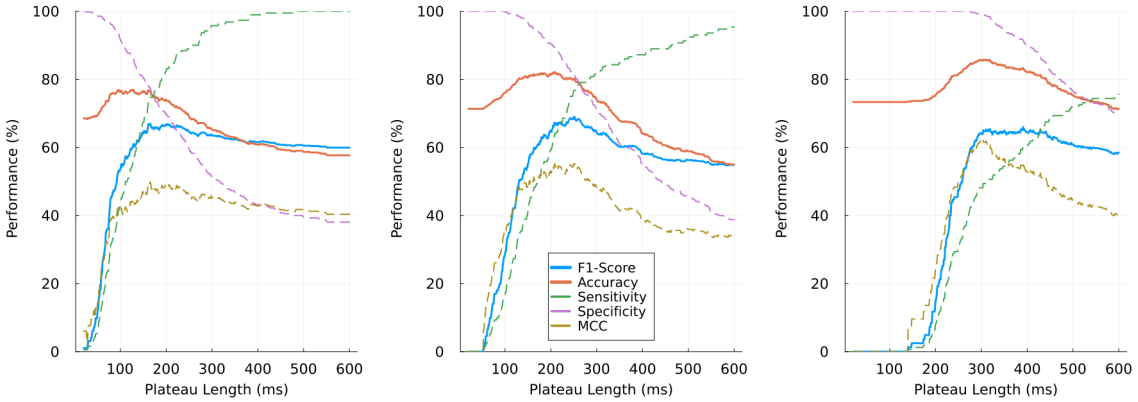


(b) unbalanced

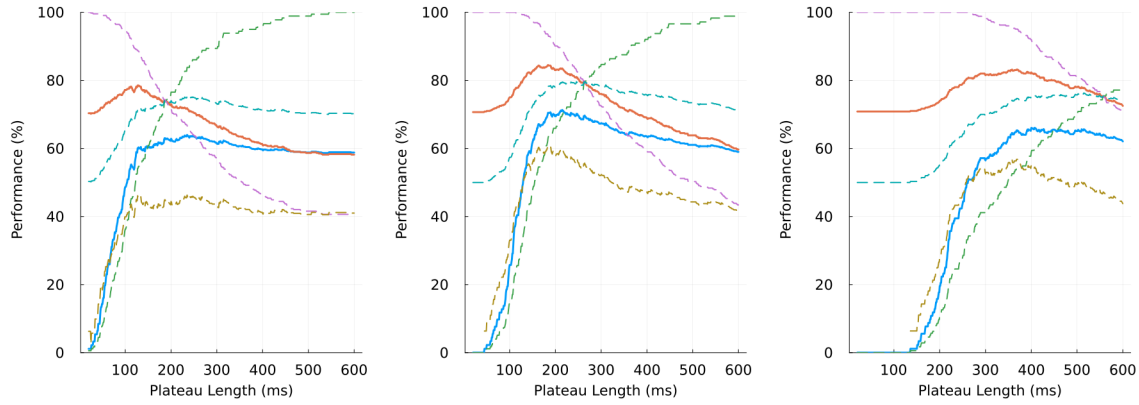


(c) extra condition

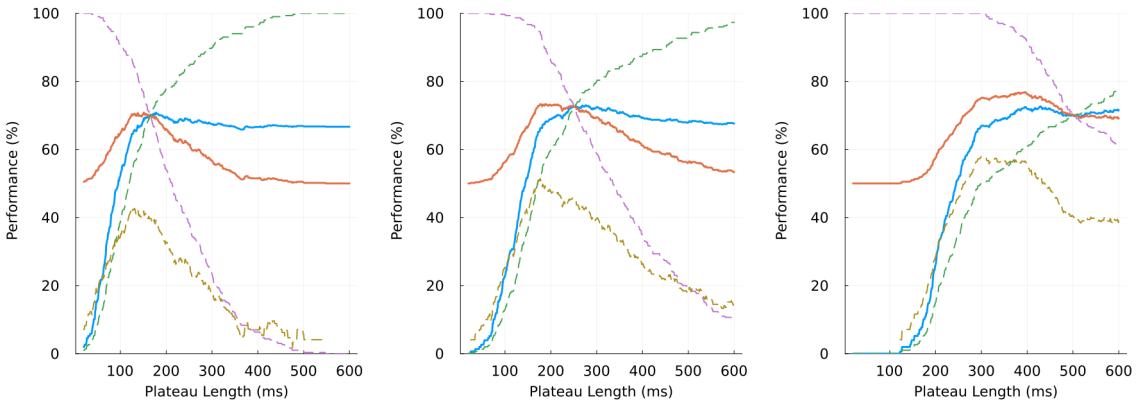
Figure A.6: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

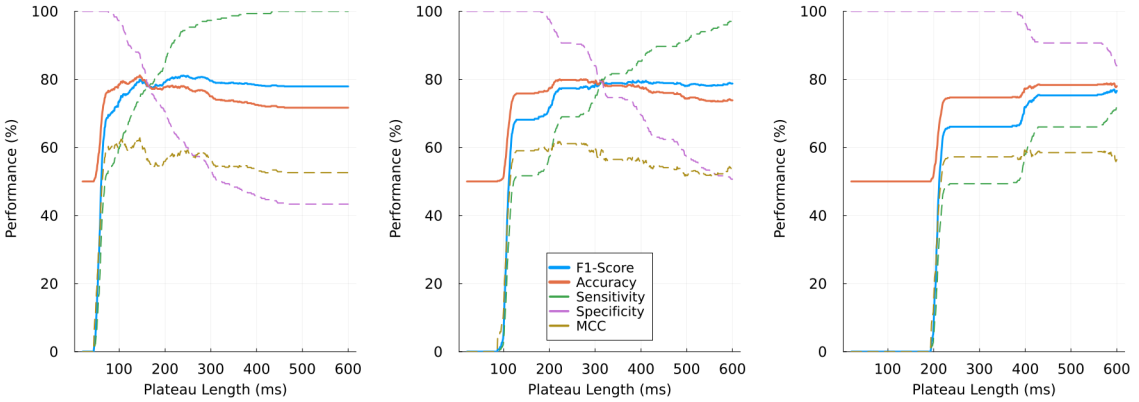


(b) unbalanced

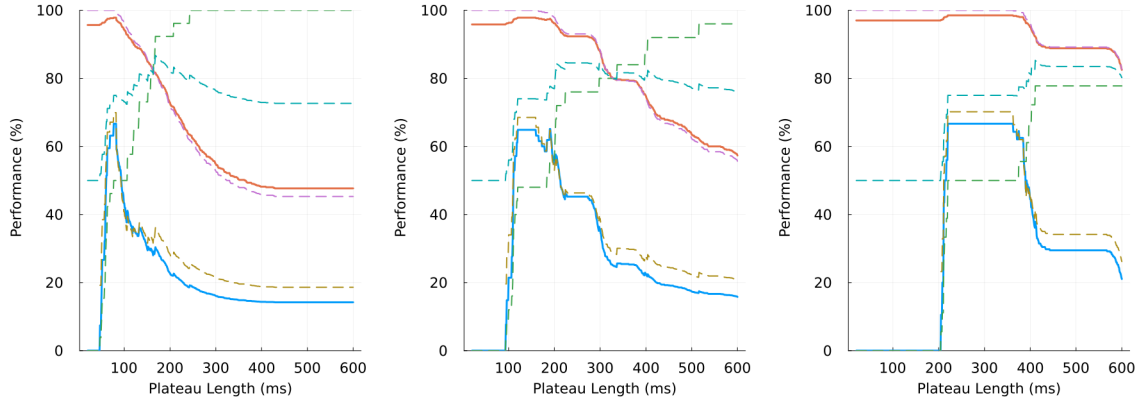


(c) extra condition

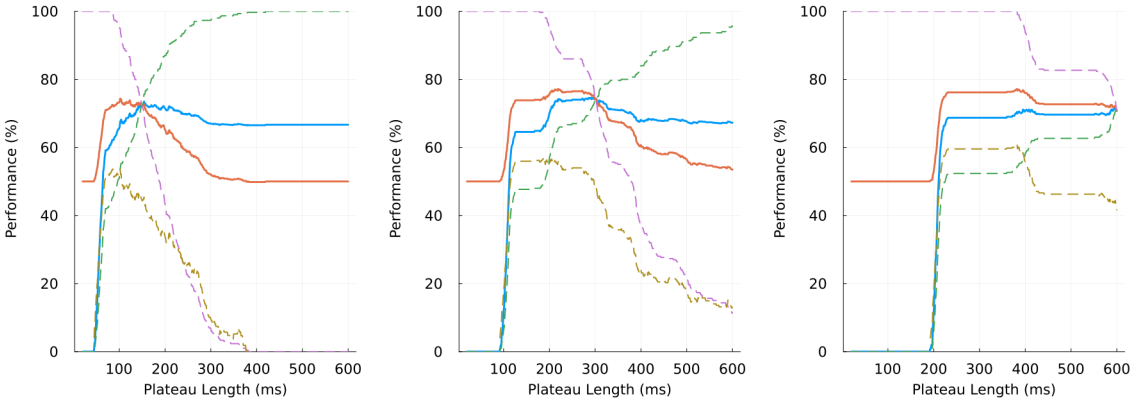
Figure A.7: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree D2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. The trials were generated from an α comprising elements [:A, :B, :C, :D]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

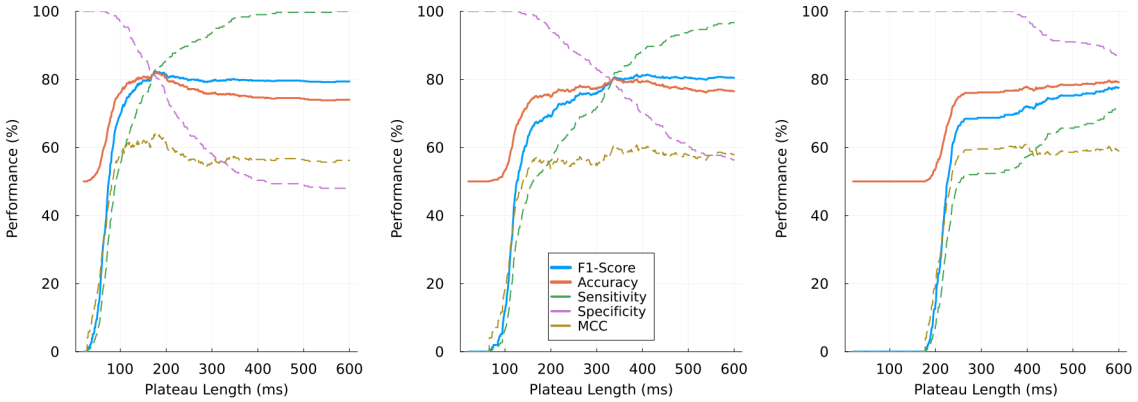


(b) unbalanced

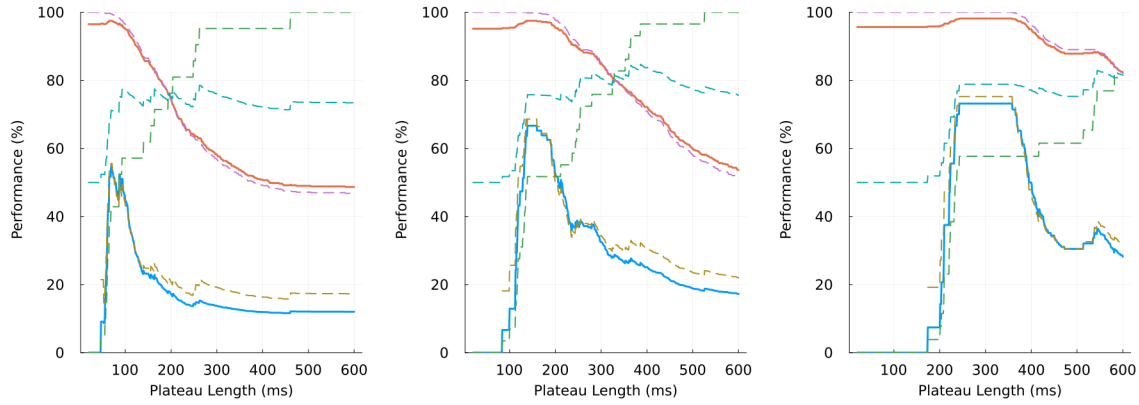


(c) extra condition

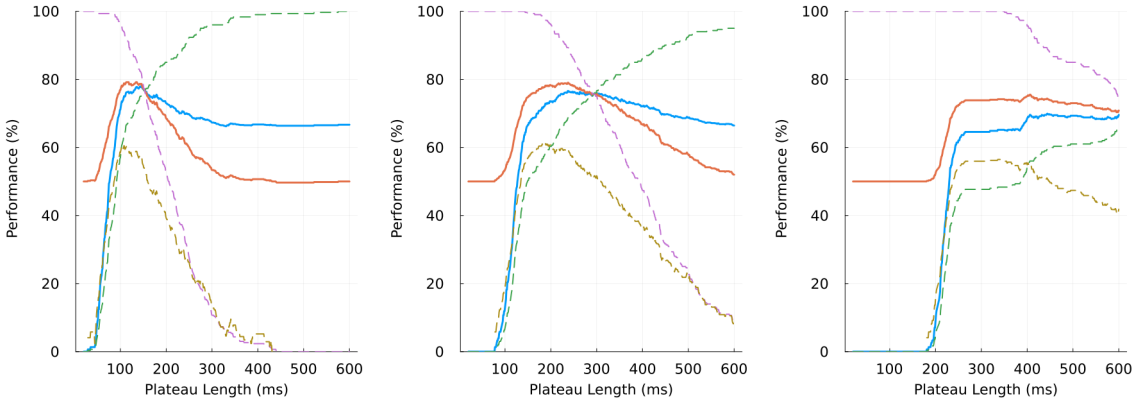
Figure A.8: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

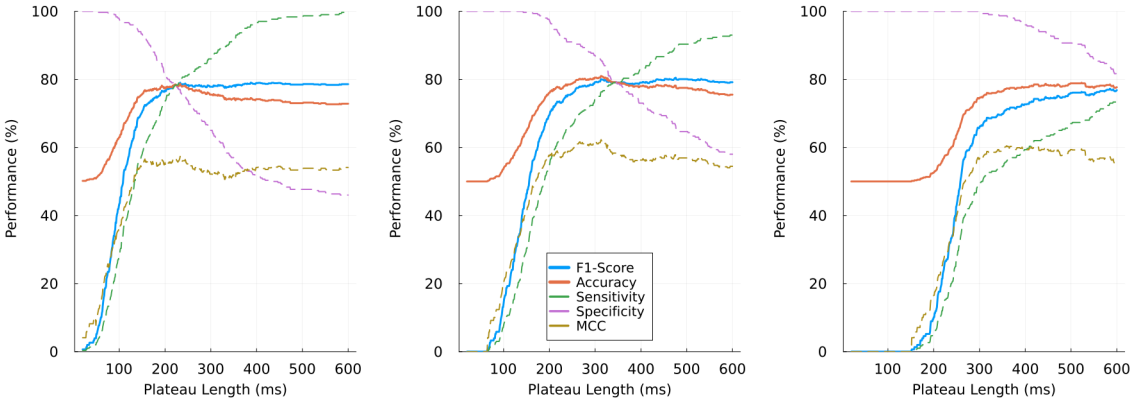


(b) unbalanced

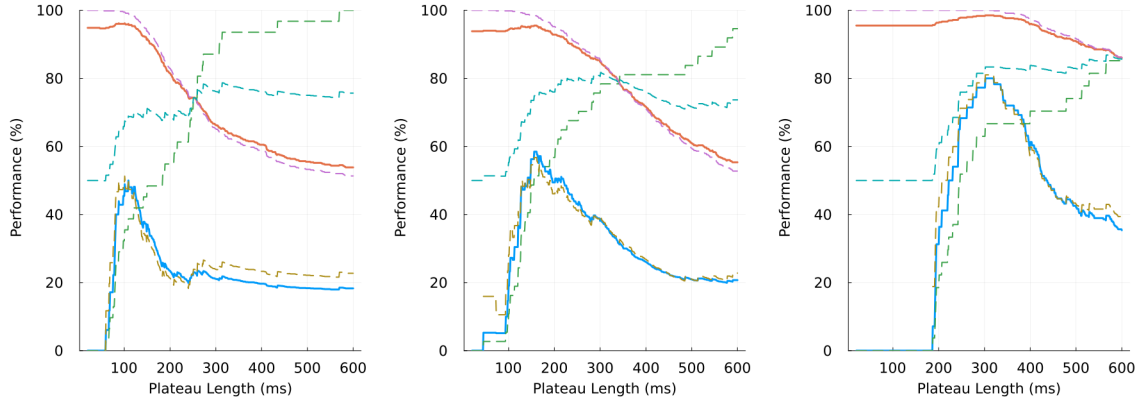


(c) extra condition

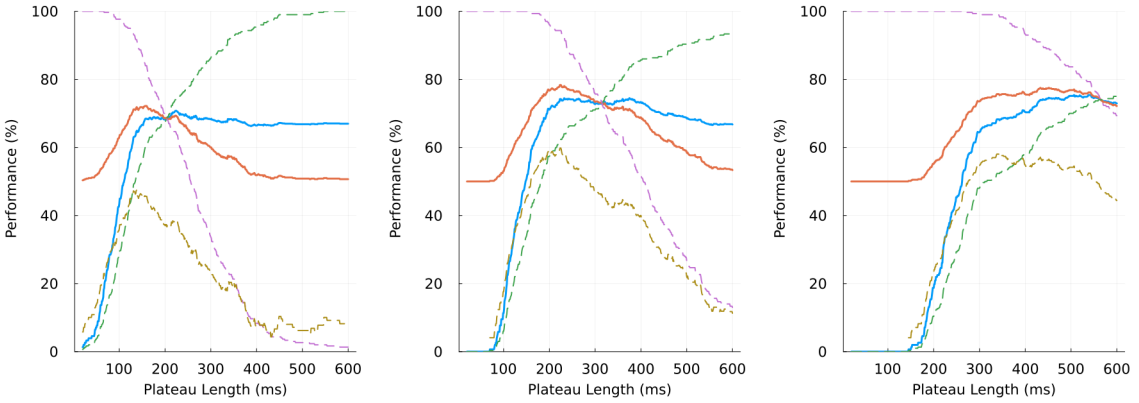
Figure A.9: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

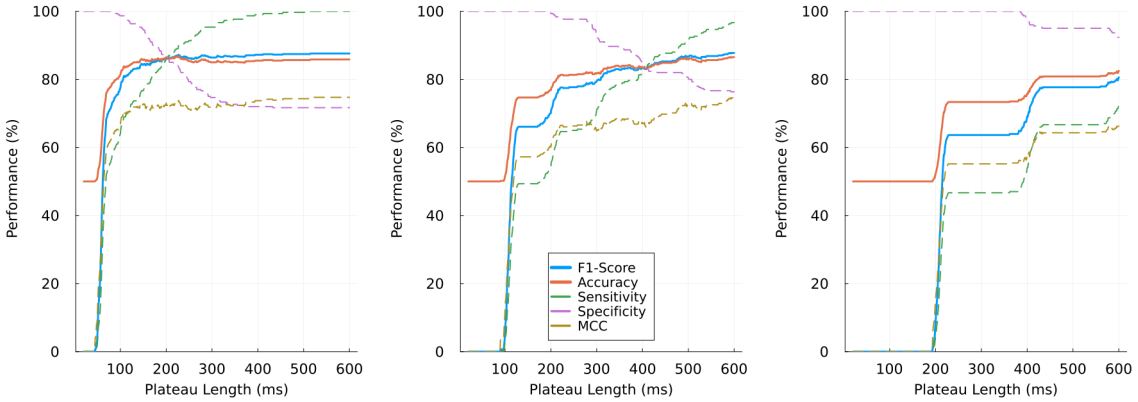


(b) unbalanced

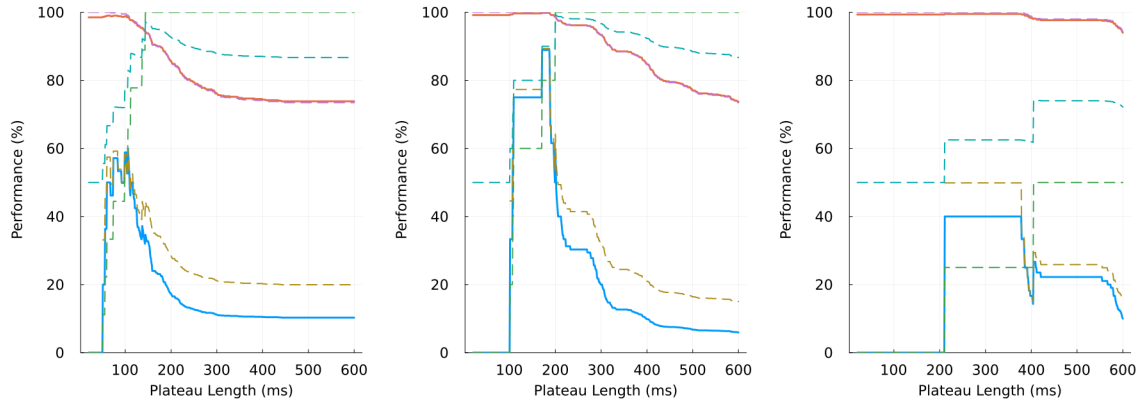


(c) extra condition

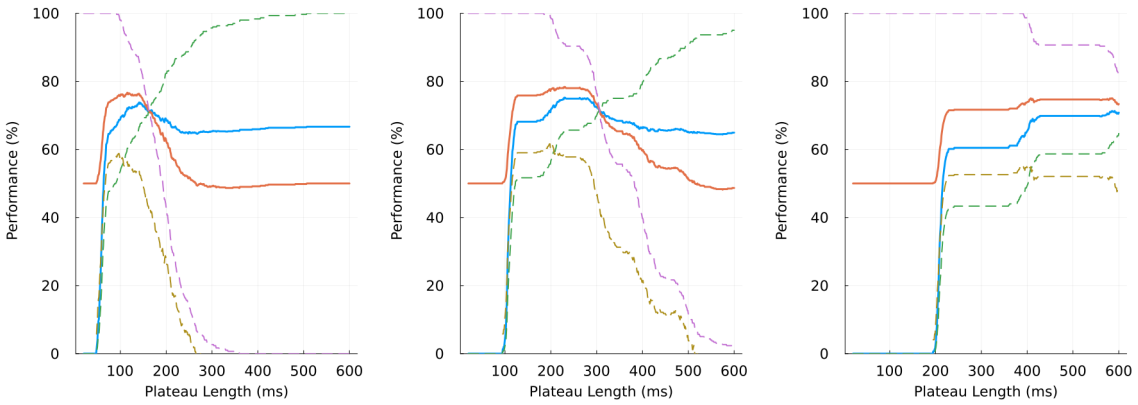
Figure A.10: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

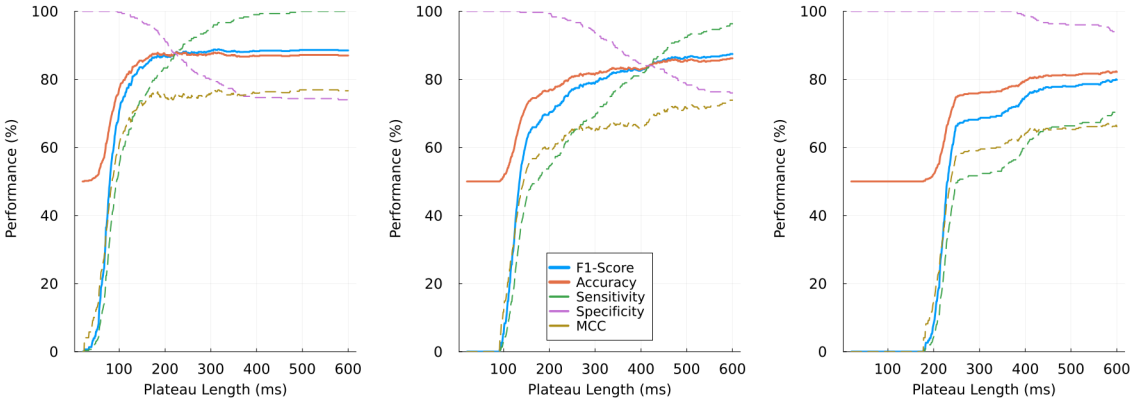


(b) unbalanced

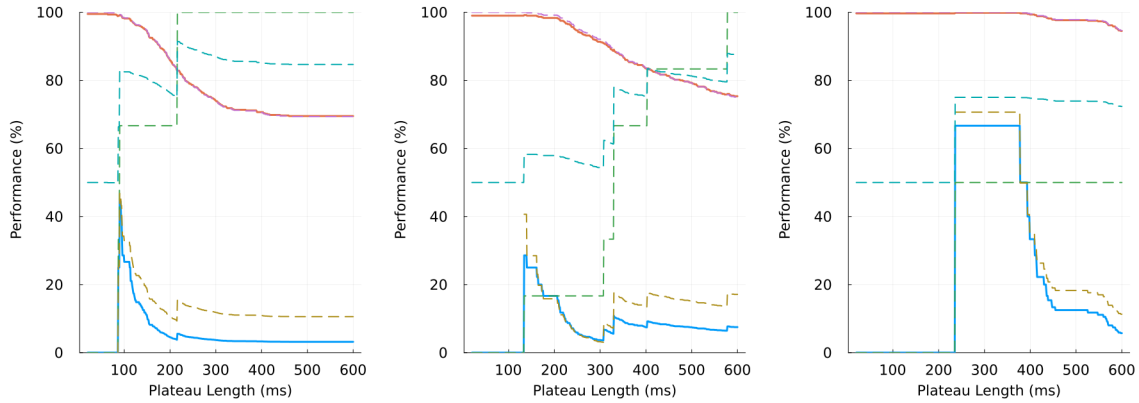


(c) extra condition

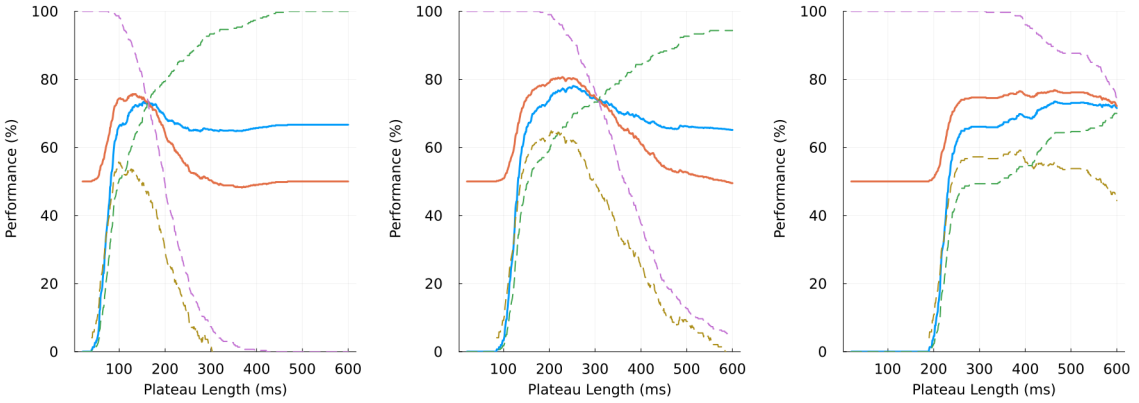
Figure A.11: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

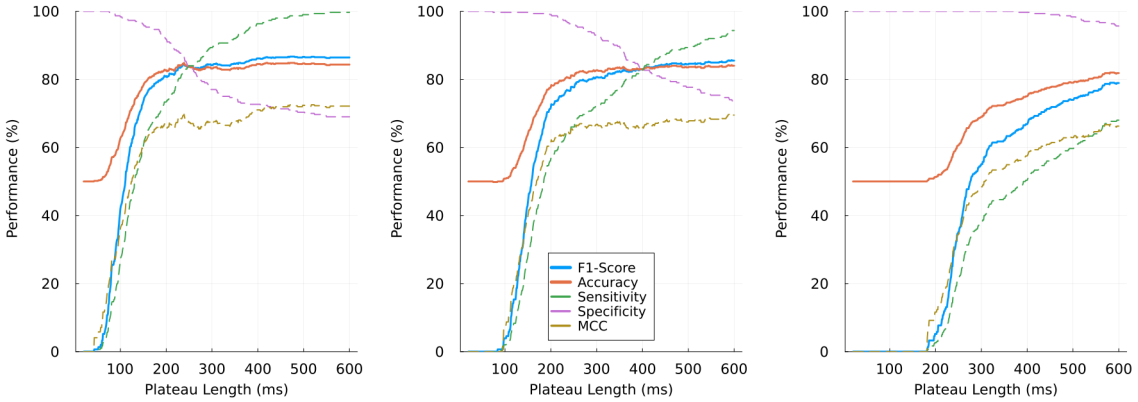


(b) unbalanced

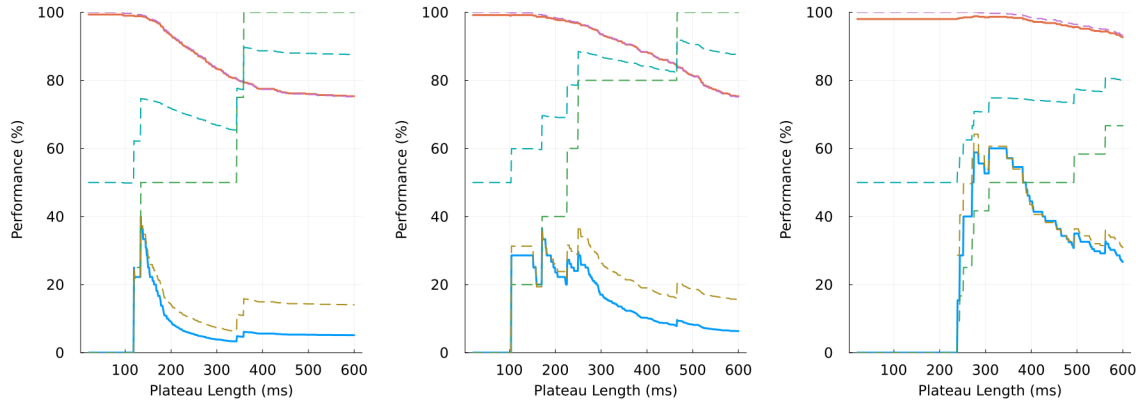


(c) extra condition

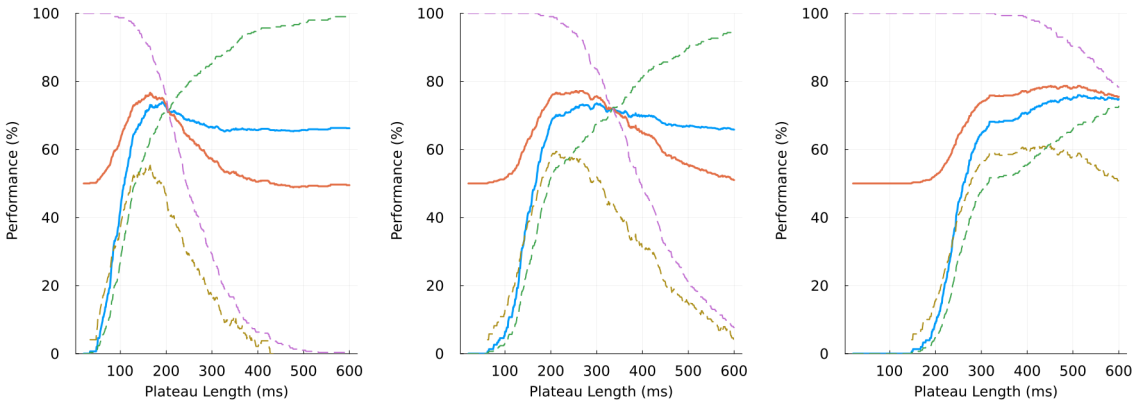
Figure A.12: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

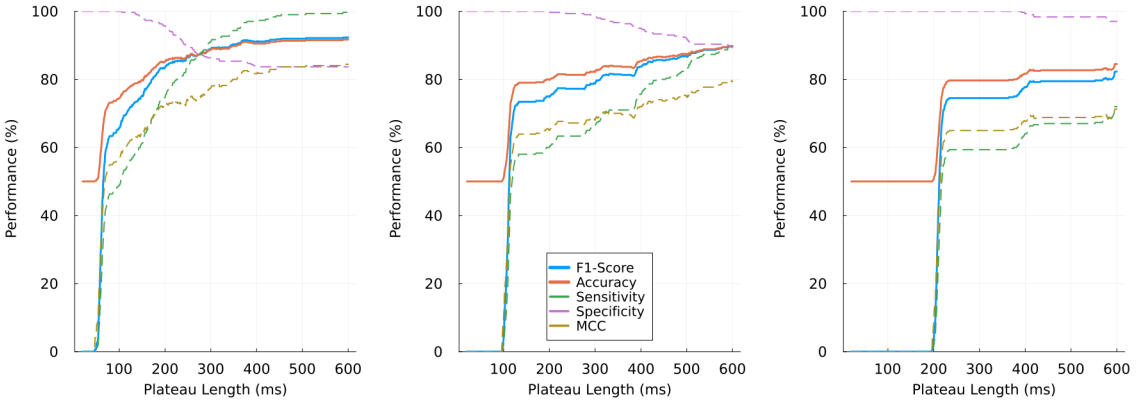


(b) unbalanced

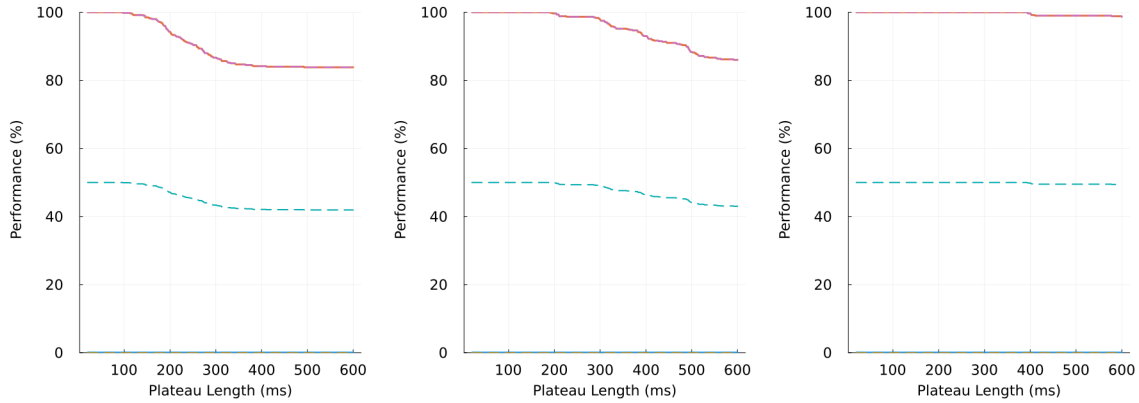


(c) extra condition

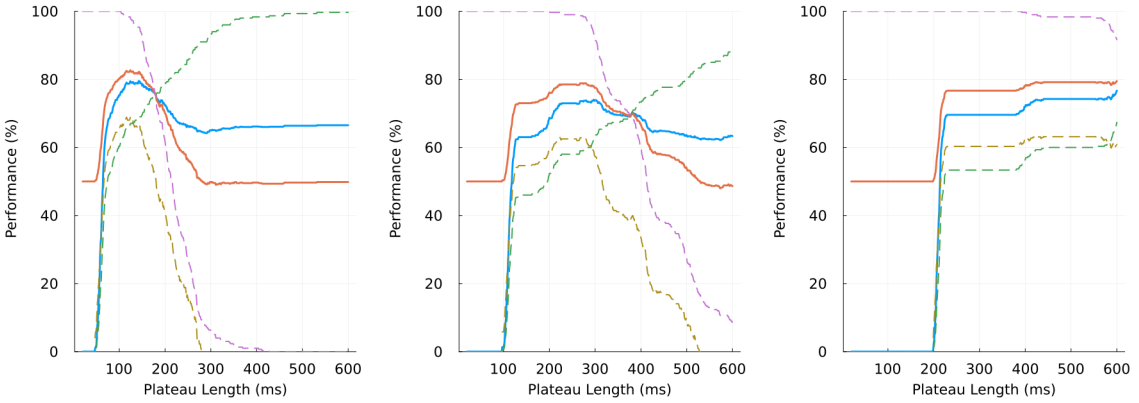
Figure A.13: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree F2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 14. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

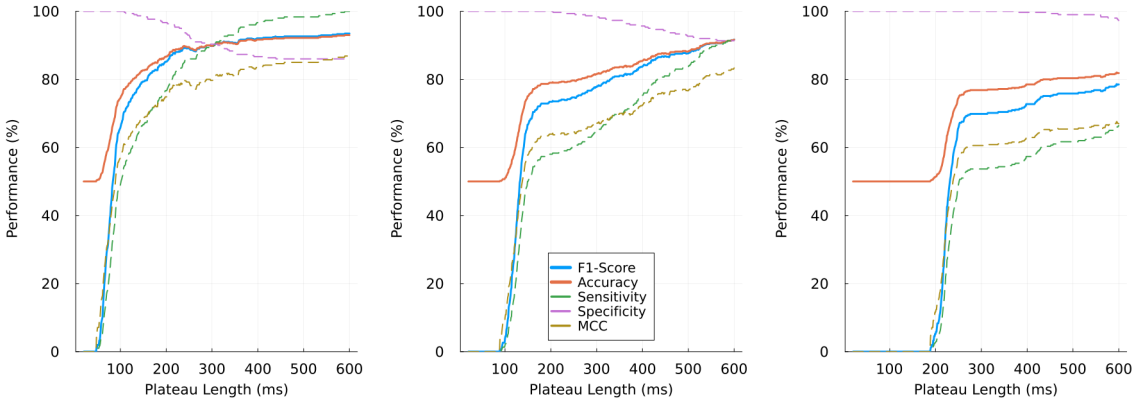


(b) unbalanced

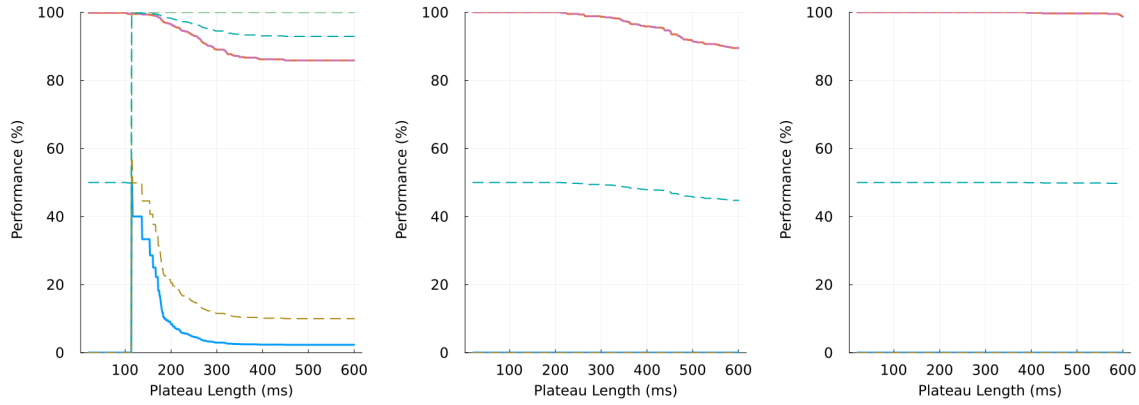


(c) extra condition

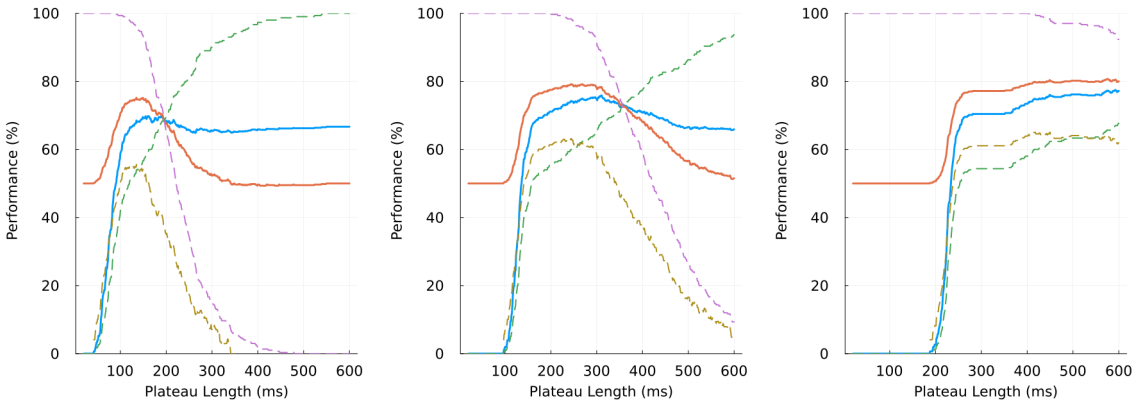
Figure A.14: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

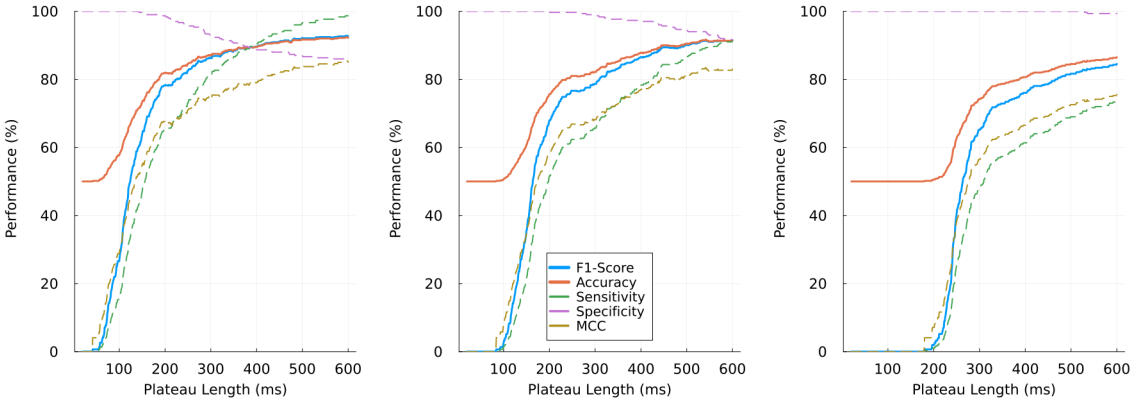


(b) unbalanced

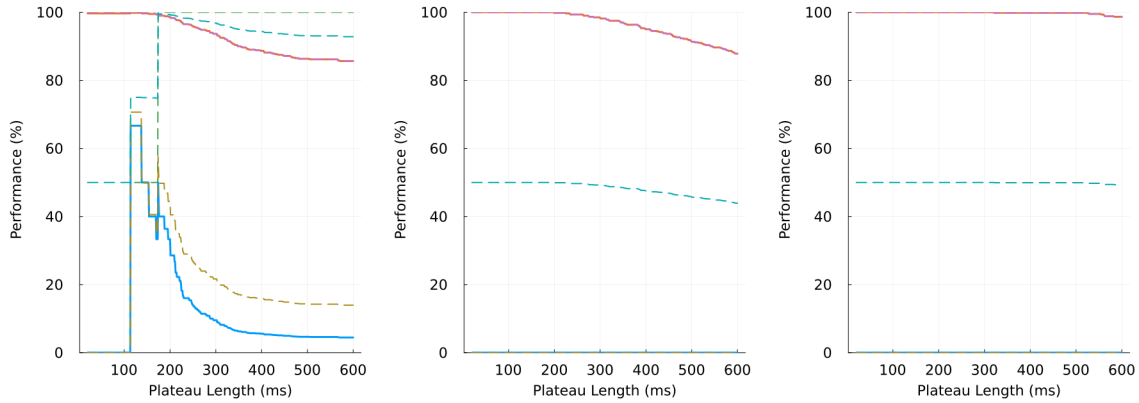


(c) extra condition

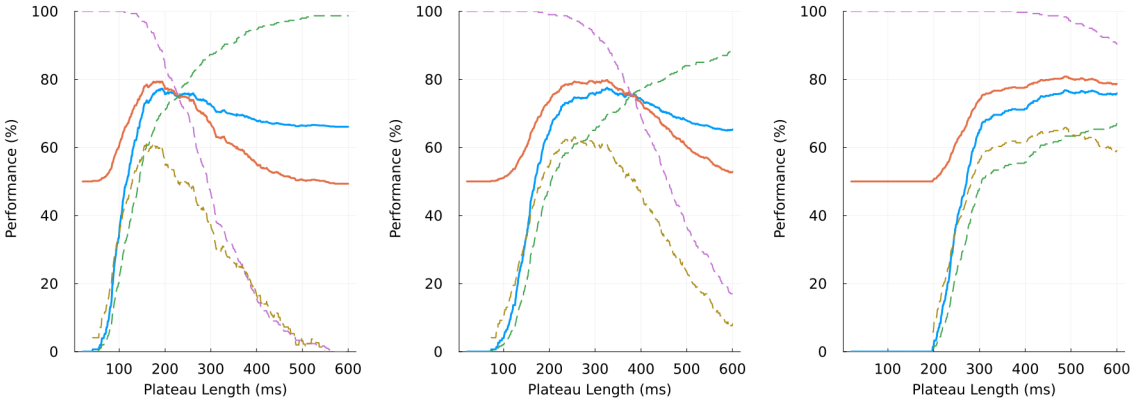
Figure A.15: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

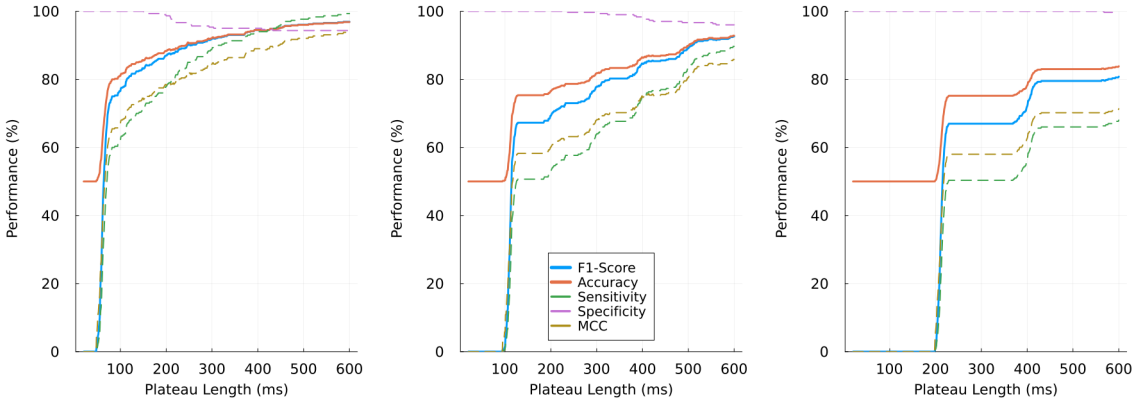


(b) unbalanced

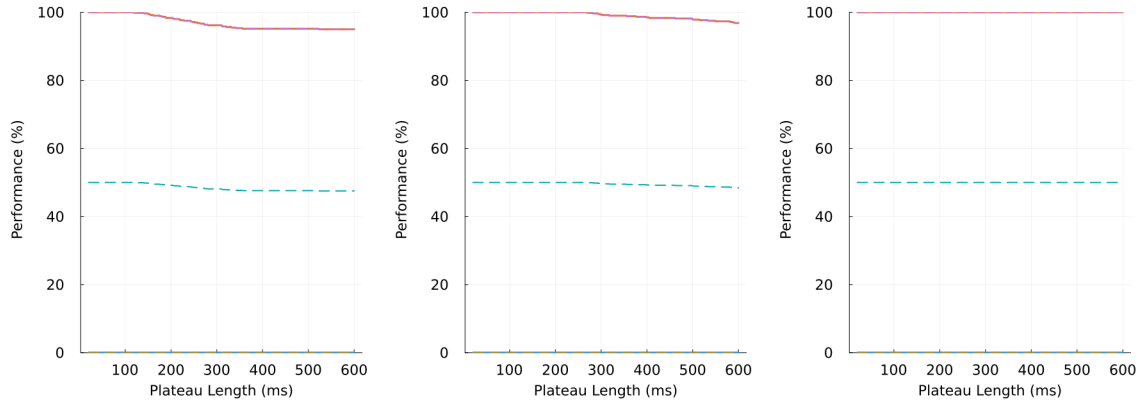


(c) extra condition

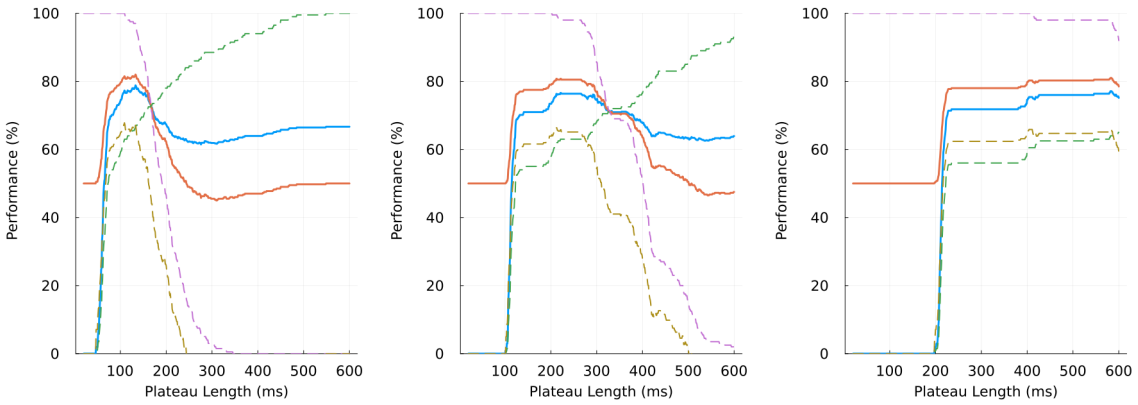
Figure A.16: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G1 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

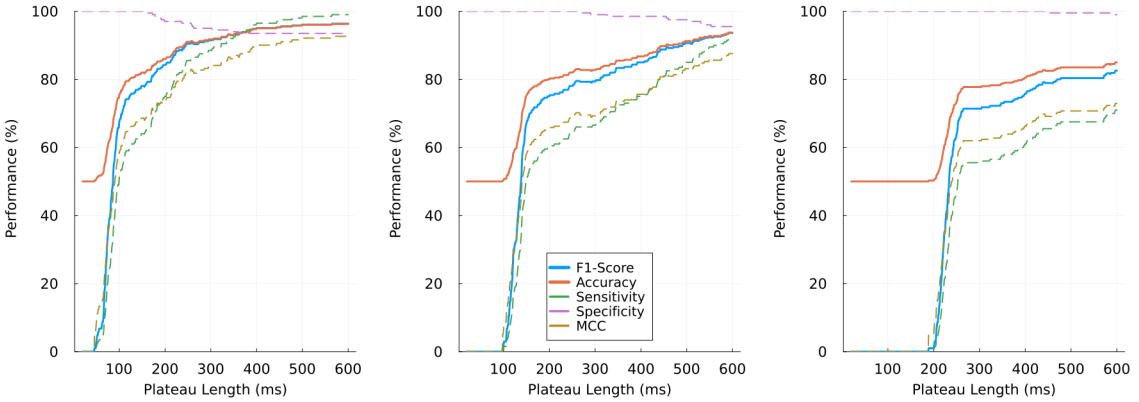


(b) unbalanced

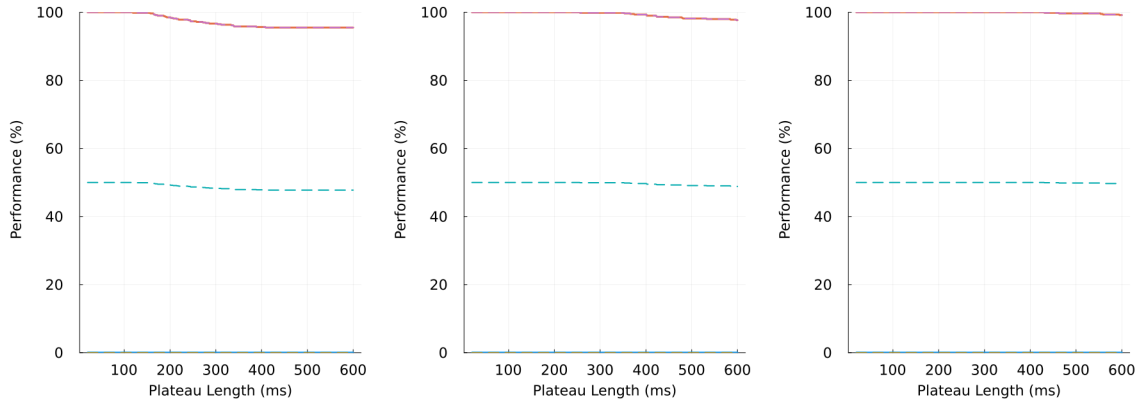


(c) extra condition

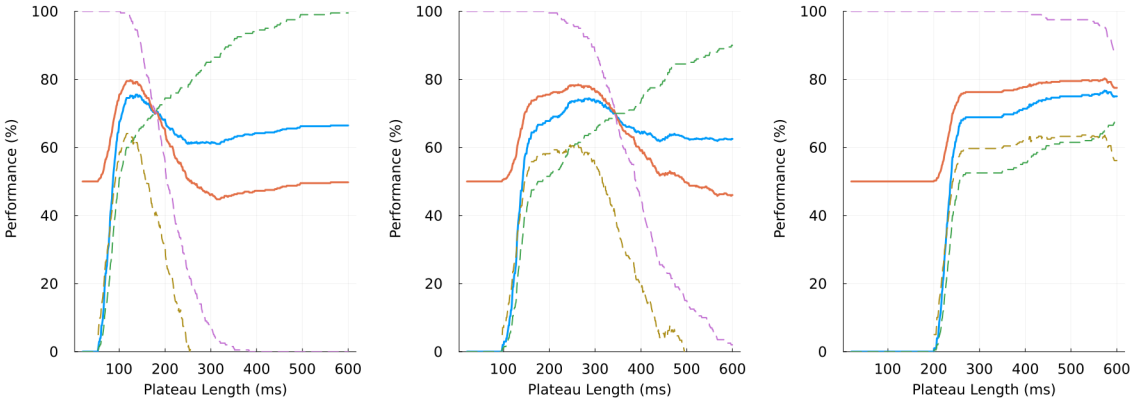
Figure A.17: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 10 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

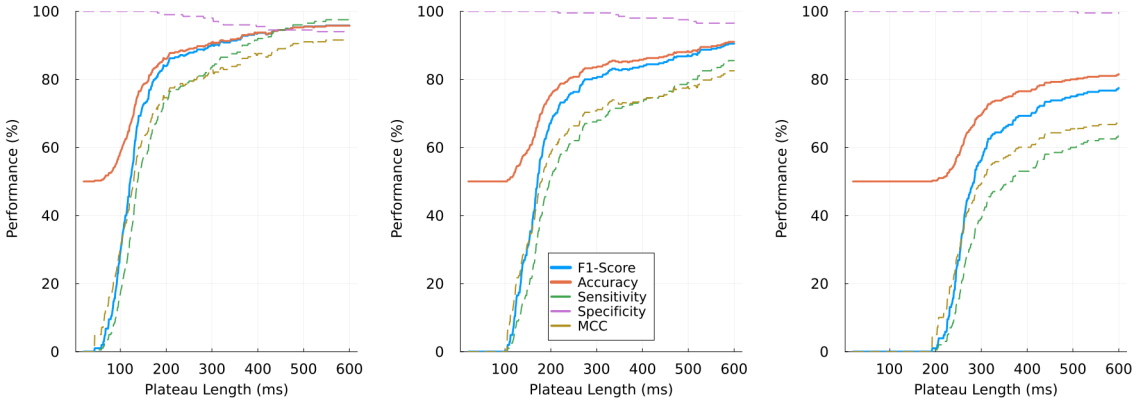


(b) unbalanced

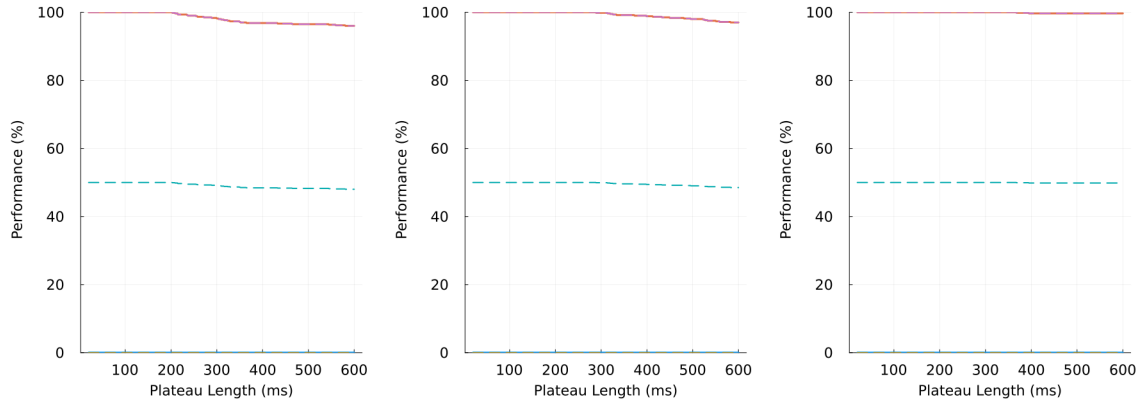


(c) extra condition

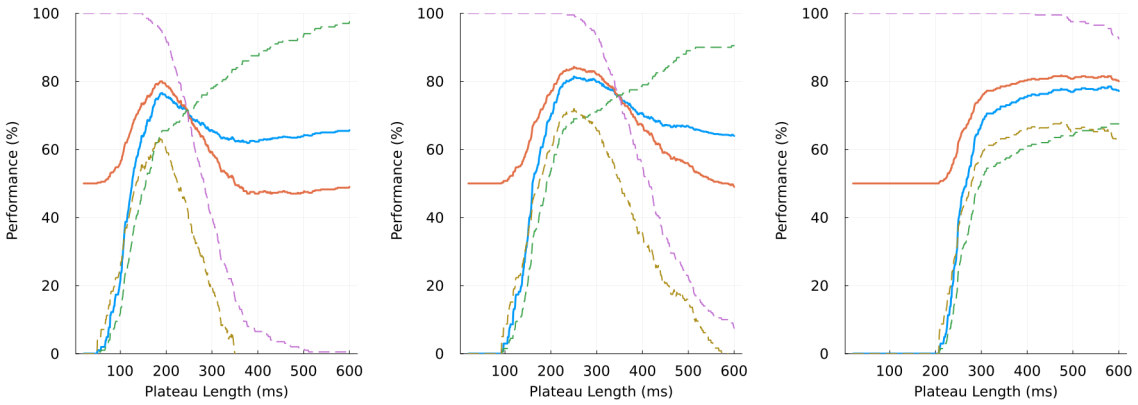
Figure A.18: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 25 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) balanced

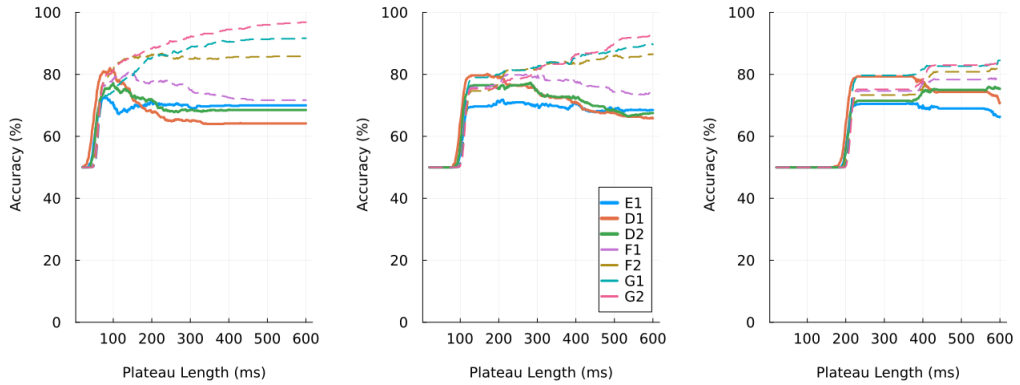


(b) unbalanced

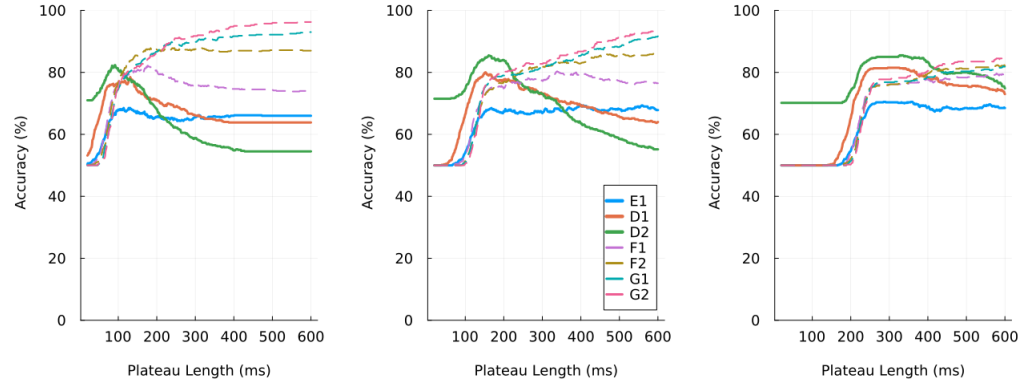


(c) extra condition

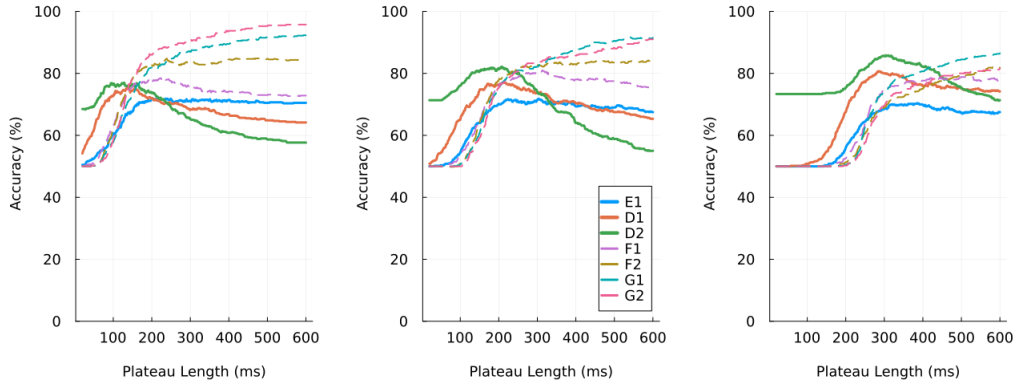
Figure A.19: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect Dendritic Tree G2 across a) balanced, b) unbalanced, and c) extra conditions. It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 18. The trials were generated from an α comprising elements [:A, :B, :C, :D, :E, :F, :G, :H]. Timing parameters followed a Gaussian distribution with a σ of 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

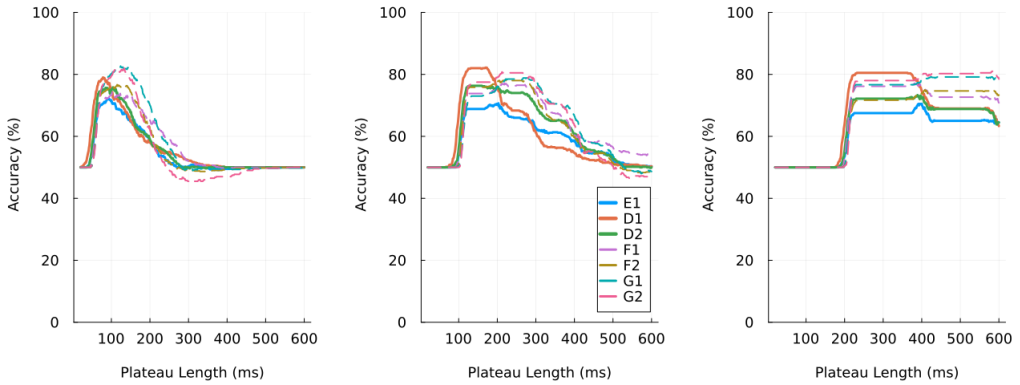


(b) $\sigma = 25$

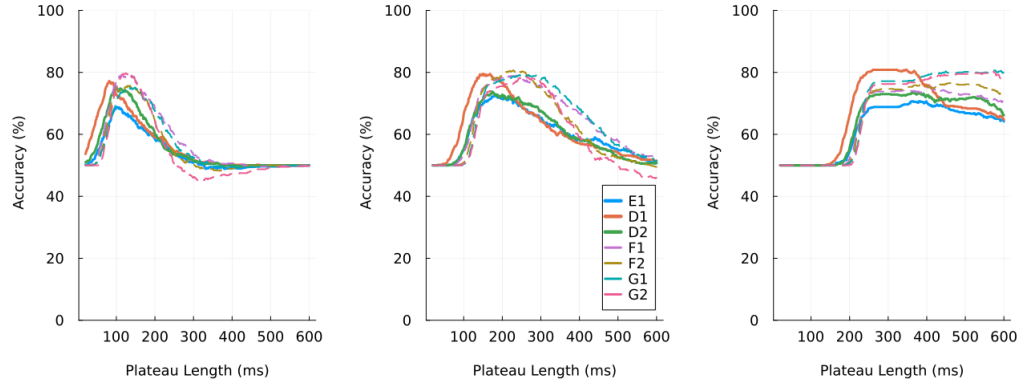


(c) $\sigma = 50$

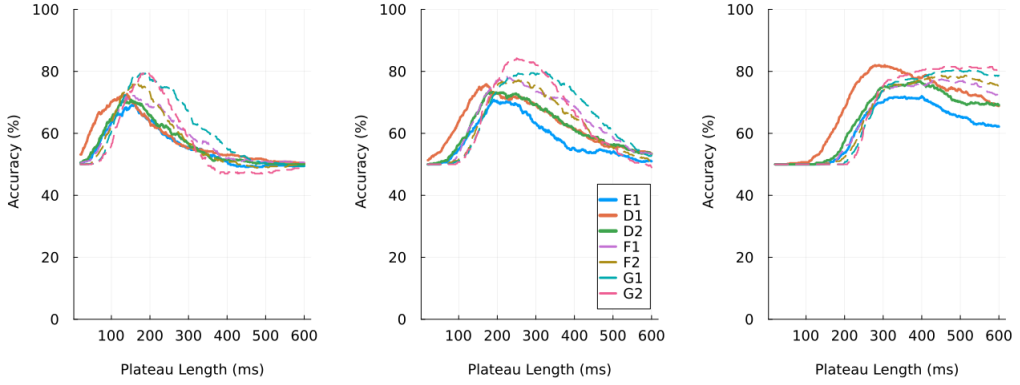
Figure A.20: The figure illustrates the accuracy performances of the naive classifier in its attempt to detect all Dendritic Trees across the balanced condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

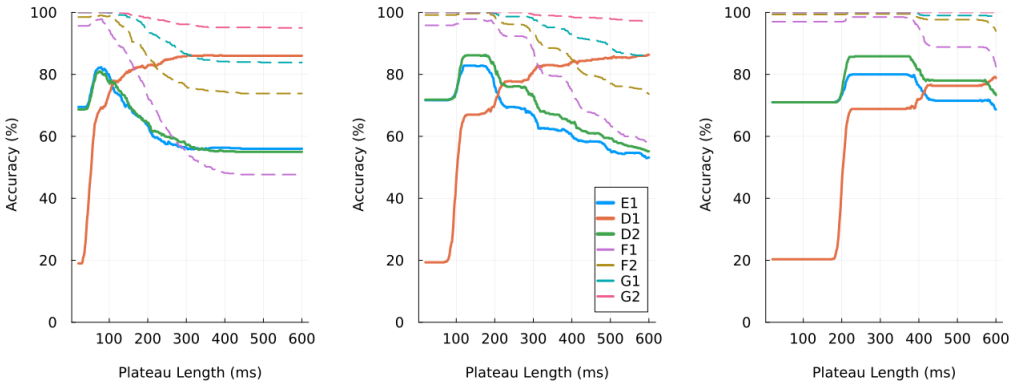


(b) $\sigma = 25$

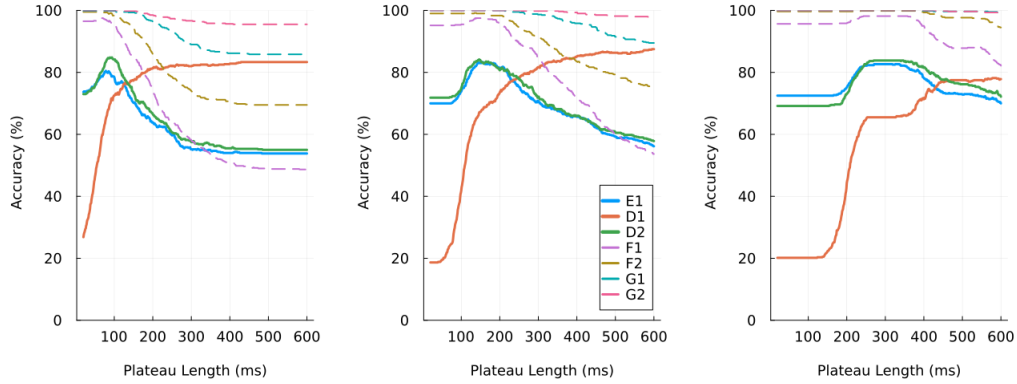


(c) $\sigma = 50$

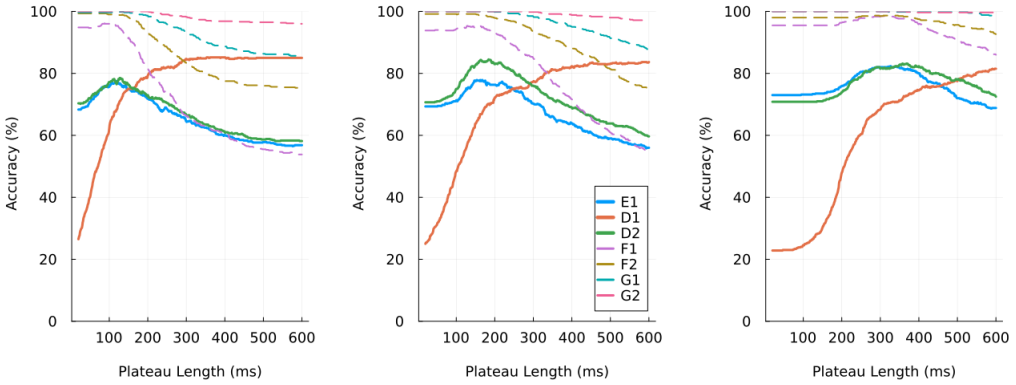
Figure A.21: The figure illustrates the accuracy performances of the naive classifier in its attempt to detect all Dendritic Trees across the extra condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

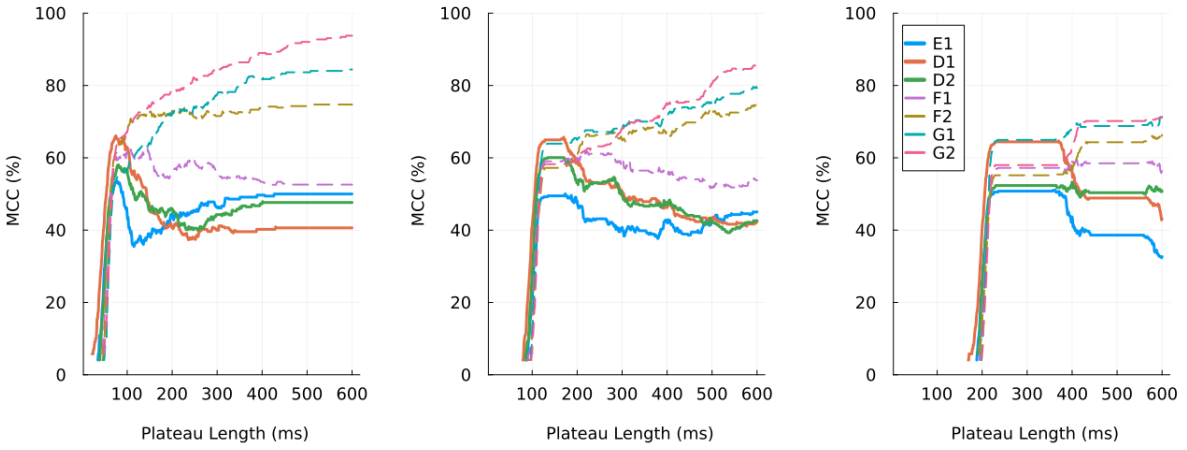


(b) $\sigma = 25$

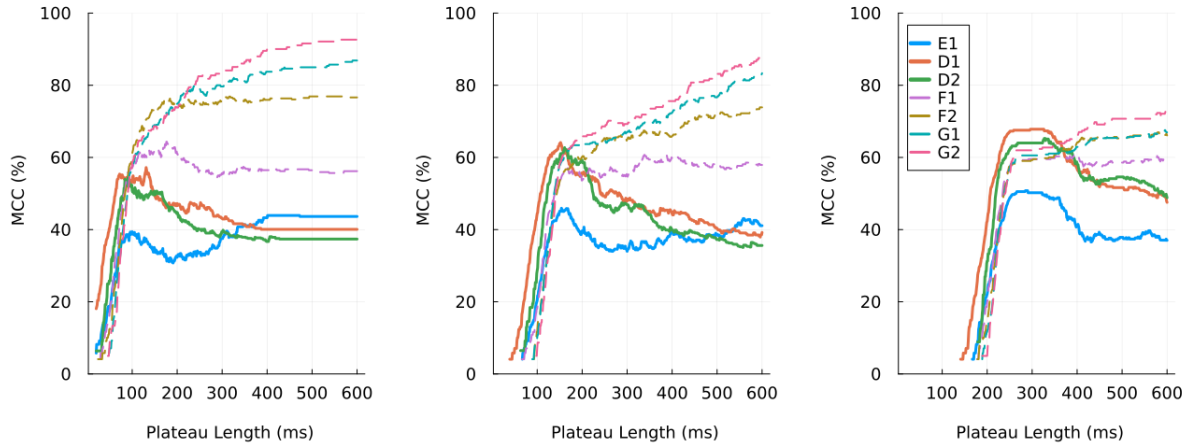


(c) $\sigma = 50$

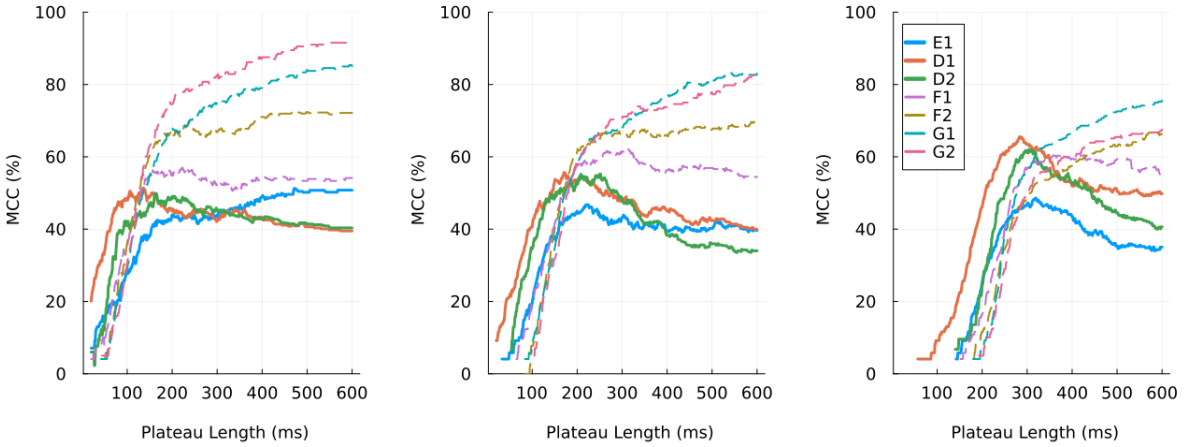
Figure A.22: The figure illustrates the accuracy performances of the naive classifier in its attempt to detect all Dendritic Trees across the unbalanced condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

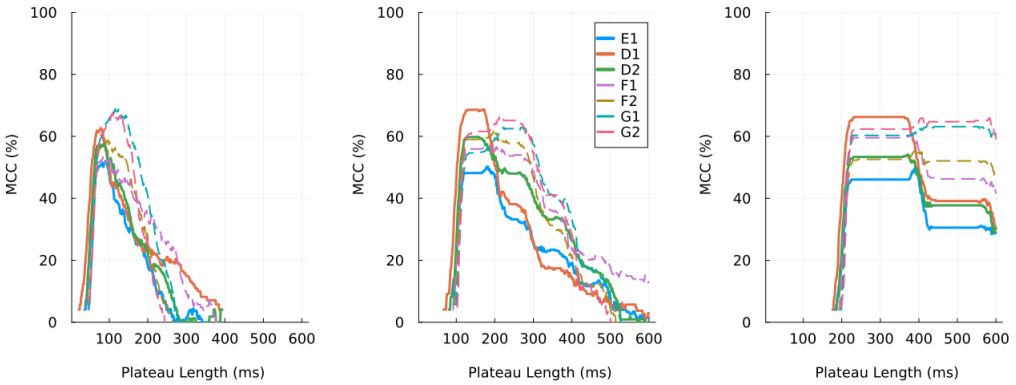


(b) $\sigma = 25$

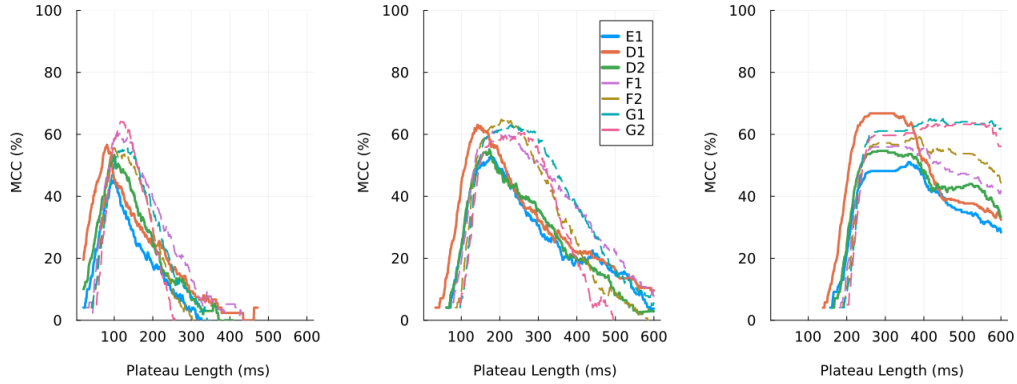


(c) $\sigma = 50$

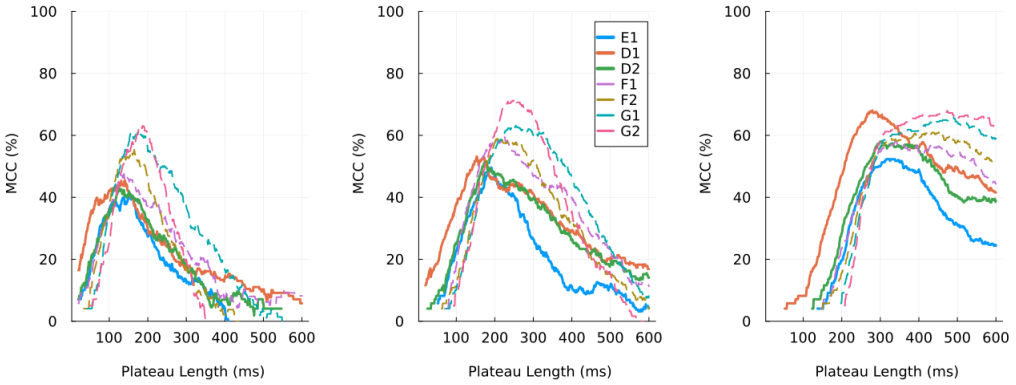
Figure A.23: The figure illustrates the MCC performances of the naive classifier in its attempt to detect all Dendritic Trees across the balanced condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

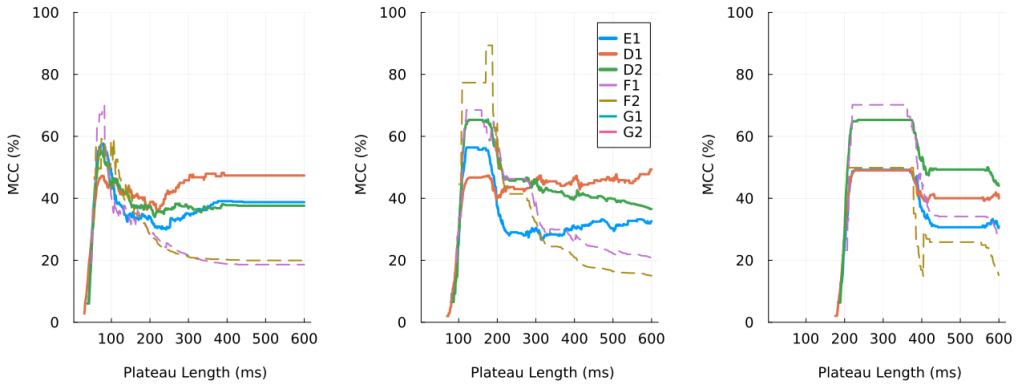


(b) $\sigma = 25$

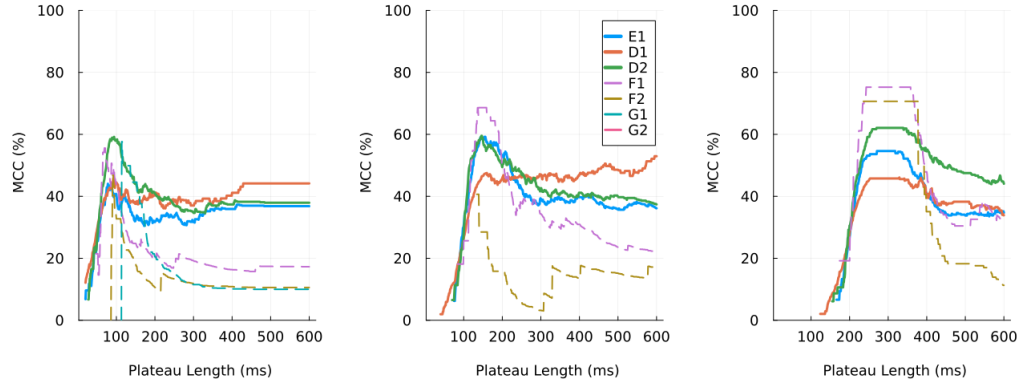


(c) $\sigma = 50$

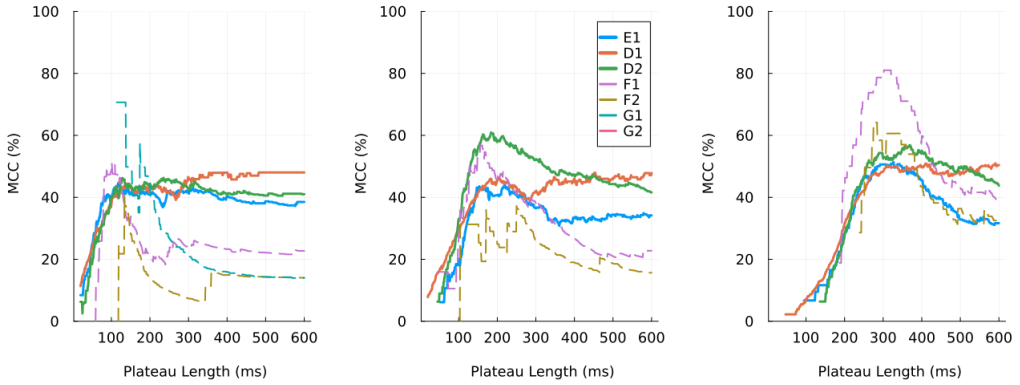
Figure A.24: The figure illustrates the MCC performances of the naive classifier in its attempt to detect all Dendritic Trees across the extra condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).



(a) $\sigma = 10$

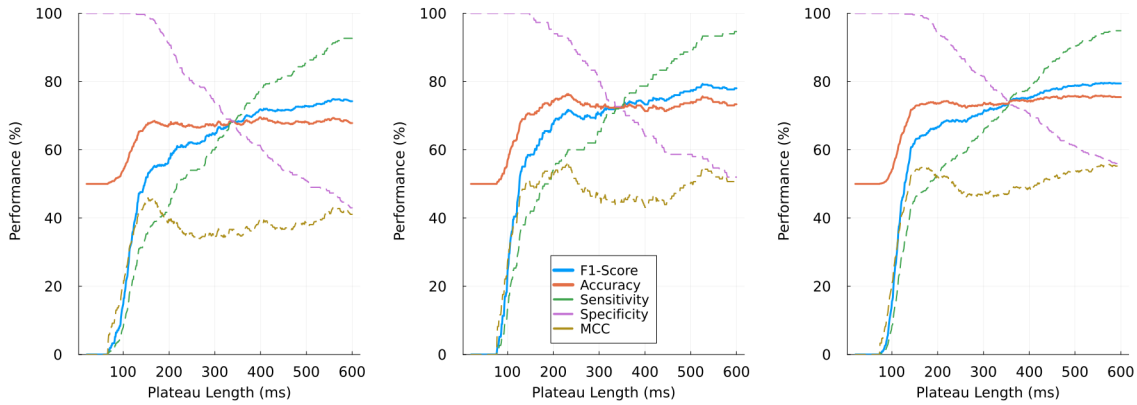


(b) $\sigma = 25$

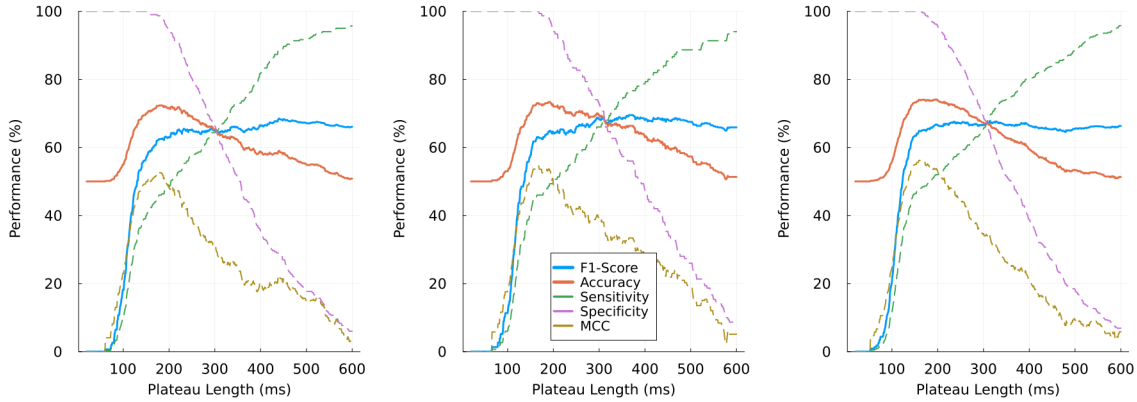


(c) $\sigma = 50$

Figure A.25: The figure illustrates the MCC performances of the naive classifier in its attempt to detect all Dendritic Trees across the unbalanced condition. The classification involved a τ ranging from 20 to 600. The sequence length and α varied for each Dendritic Tree as discussed in Section 3.2.1. Timing parameters followed a Gaussian distribution with σ values of a) 10, b) 25, and c) 50 and μ values spatially depicted as 50 (left), 100 (middle), and 200 (right).

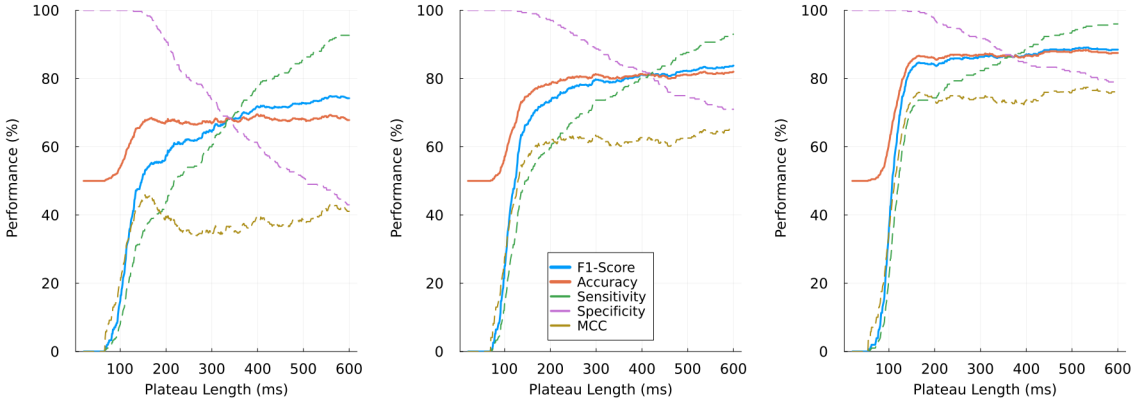


(a) balanced

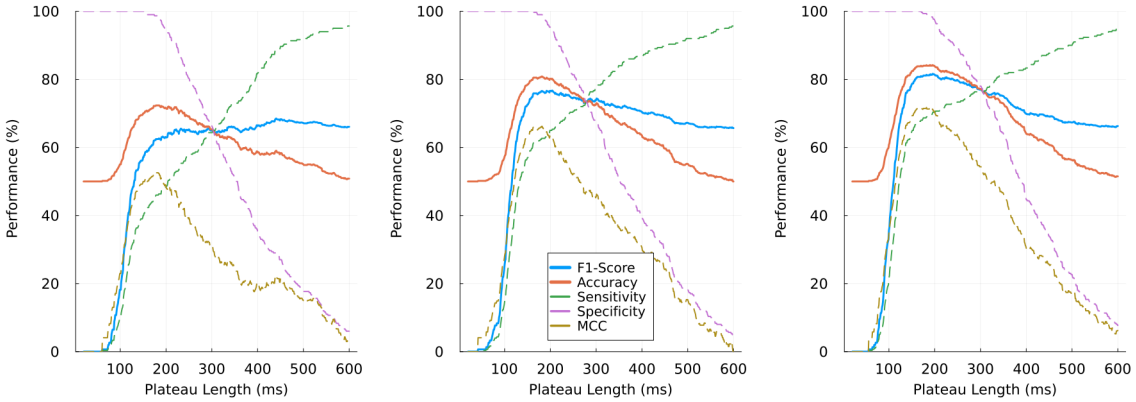


(b) extra

Figure A.26: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect the Dendritic Tree E1 across a) balanced and b) extra conditions with varying trial counts depicted as 300 (left), 600(middle) and 900 (right). It involved 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 a μ of 100.

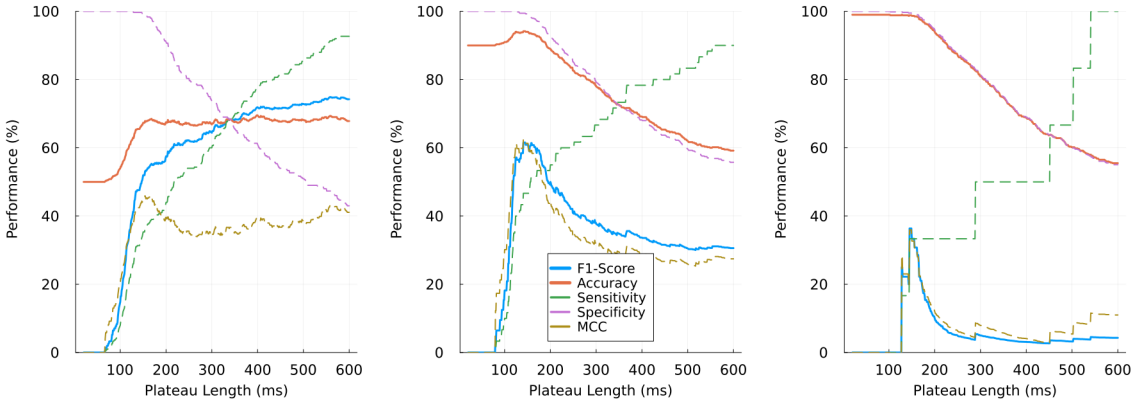


(a) balanced

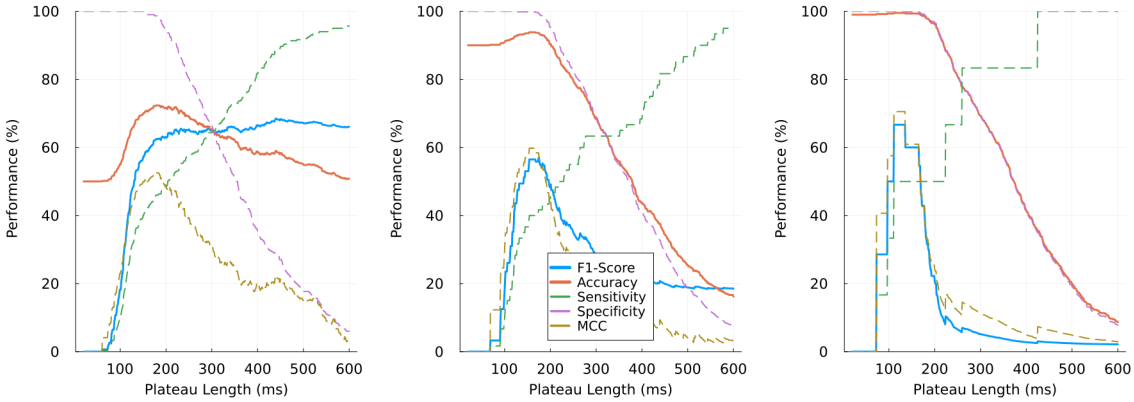


(b) extra

Figure A.27: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect the Dendritic Tree E1 across a) balanced and b) extra conditions with varying α depicted as [:A, :B, :C, :D] (left), [:A, :B, :C, :D, :E, :F] (middle) and [:A, :B, :C, :D, :E, :F, :G, :H, :I] (right). It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. Timing parameters followed a Gaussian distribution with a σ of 25 and a μ of 100.

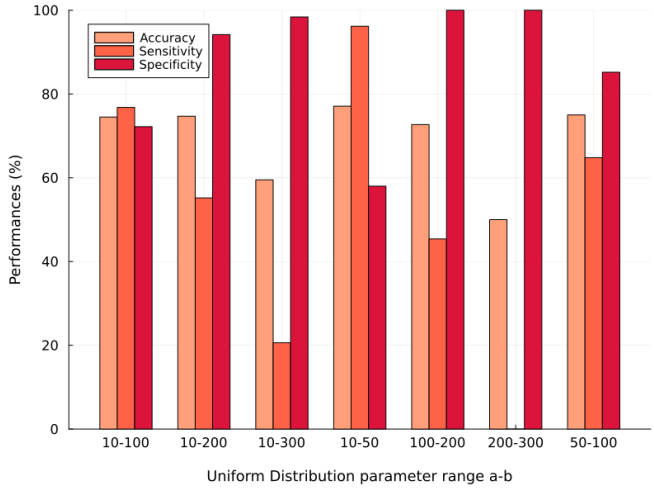


(a) balanced

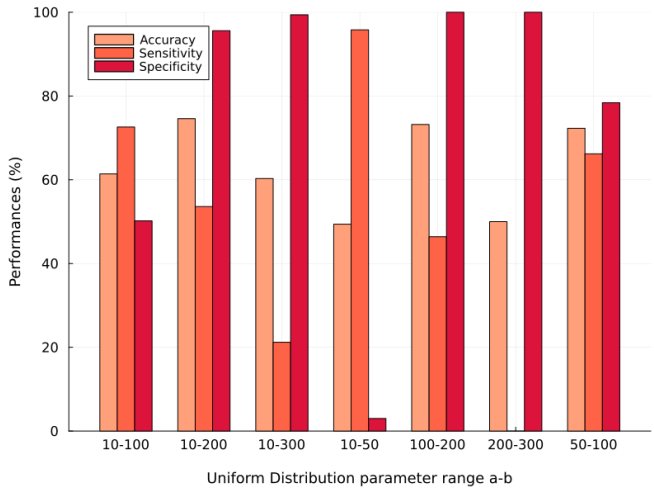


(b) extra

Figure A.28: The figure illustrates the performance evaluations of the naive classifier in its attempt to detect the Dendritic Tree E1 across a) balanced and b) extra conditions with varying negative bias depicted as 50% (left), 90% (middle) and 99% (right). It involved 600 trials with a τ ranging from 20 to 600 and a sequence length of 10. Timing parameters followed a Gaussian distribution with a σ of 25 and a μ of 100.



(a) Balanced



(b) Extra condition

Figure A.29: The figure illustrates the performances (Accuracy, Sensitivity, Specificity) of the naive classifier trying to classify Dendritic Tree E1 with a) balanced and b) extra conditions. It involved a trial count of 1000 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.