MEMORY NETWORKS AS NEUROCOMPUTATIONAL MODELS OF COGNITIVE FUNCTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

SINAN ONUR ALTINUÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF COGNITIVE SCIENCE

JULY 2024

Memory Networks as Neurocomputational Models of Cognitive Functions

submitted by **SINAN ONUR ALTINUÇ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Cognitive Science , Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç _____
Dean, **Graduate School of Informatics**

Assoc. Prof. Dr. Barbaros Yet _____
Head of Department, **Cognitive Science**

Assoc. Prof. Dr. Murat Perit Çakır _____
Supervisor, **Cognitive Science, METU**

Examining Committee Members:

Assoc. Prof. Dr. Barbaros Yet
Cognitive Science, METU _____

Assoc. Prof. Dr. Murat Perit Çakır
Cognitive Science, METU _____

Assoc. Prof. Dr. Cengiz Acartürk
Cognitive Science, Jagiellonian University _____

Assist. Prof. Dr. Umut Özge
Cognitive Science, METU _____

Assist. Prof. Dr. Burcu A. Ürgen
Psychology, Bilkent University _____

Date: 16.07.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Sinan Onur ALTINUÇ

Signature :

ABSTRACT

MEMORY NETWORKS AS NEUROCOMPUTATIONAL MODELS OF COGNITIVE FUNCTIONS

ALTINUÇ, Sinan Onur

Ph.D., Department of Cognitive Science

Supervisor: Assoc. Prof. Dr. Murat Perit Çakır

July 2024, 181 pages

This thesis develops a cognitive computational framework for exploring memory networks as neurocomputational models of syntax and rule learning, with a particular focus on Neural Turing Machines (NTMs). It contrasts NTMs with recurrent neural networks such as long short-term memory (LSTM) and Legendre memory unit (LMU) architectures.

By using artificial grammars to evaluate syntactic processing capabilities and specifically assessing the rule-learning and generalization abilities of these neural architectures, it studies the role of memory mechanisms in cognitive models, particularly the role of external memory. The experiments involve training neural networks on various types of grammar, including regular, context-free grammars (CFGs) and mildly context-sensitive grammars (MCFGs), with a focus on hierarchical structures and long-distance dependencies. Comparative analysis in the thesis shows that unlike regular recurrent networks, neural networks with external memory structures such as NTM were able to learn and generalize CFGs beyond training data by learning the rules. Supported also by meta-analysis, this thesis shows that augmented memory networks can generalize grammars with MCSG properties with a few exceptions.

This capacity of the memory networks is compared with linguistic, and neuroscientific studies to understand the structural biases that might be required for these tasks and their potential neural correlates in the human brain. This thesis suggests that the ability to represent and store explicit representations in an external memory might be a key factor for rule-based symbolic operations on neural networks with implications for neuroscience and AI.

Keywords: Memory Networks, Rule Learning, Generalization, Computational Models of Mind, Working Memory

ÖZ

KOGNİTİF FONKSİYONLARIN SİNİRBİLİMSEL MODELLERİ OLARAK BELLEK AĞLARI

ALTINUÇ, Sinan Onur

Doktora, Bilişsel Bilimler Bölümü

Tez Yöneticisi: Doç. Dr. Murat Perit Çakır

Temmuz 2024, 181 sayfa

Bu tez, bellek ağlarını sözdizim ve kural öğreniminin nörobilgisayar modelleri olarak incelemek için bir bilişsel hesaplama çerçevesi geliştirmektedir. Özellikle Nöral Turing Makineleri (NTMs) üzerine odaklanılarak, bu yapıların uzun kısa süreli bellek (LSTM) ve Legendre bellek birimi (LMU) gibi tekrarlayan sinir ağlarıyla karşılaştırılması yapılmaktadır.

Yapay gramerler kullanarak bu sinirsel mimarilerin sözdizimsel işlem yeteneklerini ve özellikle kural öğrenme ve genelleme becerilerini değerlendirerek, bilişsel modellerdeki bellek mekanizmalarının, özellikle de harici belleklerin rolü incelenmiştir. Deneyler, sinir ağlarının hiyerarşik yapılar ve uzun mesafe bağımlılıklarına odaklanarak, düzenli (regullar), context-free gramerler (CFG'ler) ve mildly context-sensitive gramerler (MCFG'ler) gibi çeşitli gramer türlerinde eğitilmesini içermektedir. Tezde yapılan karşılaştırmalı analiz, harici bellek yapıları olan NTM'lerin, düzenli tekrarlayan ağların aksine, kuralları öğrenerek CFG'leri eğitim verilerinin ötesinde öğrenip genelleyebildiğini göstermektedir. Ayrıca, meta analizle destekleyerek, bu tez, bazı istisnalar dışında, geliştirilmiş bellek ağlarının MCSG özelliklerine sahip gramerleri genelleyebildiğini göstermektedir.

Bellek ağlarının bu kapasitesi, bu görevler için gerekli olabilecek yapısal önyargıları ve insan beynindeki olası sinirsel karşılıklarını anlamak amacıyla dilbilimsel ve sinirbilimsel çalışmalarla karşılaştırılmaktadır. Bu tez, harici bir bellekte açık temsillerin temsil edilmesi ve saklanabilmesinin, sinir ağlarında kural tabanlı sembolik işlemler için anahtar bir faktör olabileceğini ve bunun sinirbilim ve yapay zeka için önemli sonuçlar doğurabileceğini önermektedir.

Anahtar Kelimeler: Hafıza Ağları, Kural Öğrenme, Zihnin Hesaplamasal Modelleri, Çalışma Belleği

To all those who laid a brick in the way to understanding intelligence.

ACKNOWLEDGMENTS

First and foremost, I extend my gratitude to my Supervisor, Assoc. Prof. Dr. Murat Perit Çakır, for his guidance and support throughout the journey of this research. His expertise in the field, his positive outlook and his ability to inspire optimism, have been instrumental in shaping this thesis.

I would like to express my heartfelt thanks to my family for their unconditional love, patience, and encouragement. Their belief in my abilities and constant support have been the pillars of my strength and resilience during this academic pursuit.

I am also immensely grateful to my friends and colleagues at the department. The stimulating discussions we shared on various topics in cognitive science have significantly enriched my perspective and understanding of the subject. These interactions have not only been intellectually rewarding but have also provided a sense of camaraderie and support that I deeply cherish.

Finally, I am mostly deeply thankful to who I consider closest to my mind and to my hearth that help me mentally and physically to all ups and downs.

This journey would not have been the same without the collective wisdom, encouragement, and support of these individuals. I am profoundly thankful to each one of them for their contributions to my academic and personal growth.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ESM	Experience Sampling Method
GLMM	Generalized Linear Mixed Model
RL	Reinforcement Learning
NTM	Neural Turing Machine
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
LMU	Legendre Memory Unit
GRU	Gated Recurrent Unit
PFC	Prefrontal Cortex
HPC	Hippocampus
EM	Episodic Memory
WM	Working Memory
PDA	Push-down Automata
UI	User Interface
LLM	Large Language Model
CFG	Context-Free Grammar
CSG	Context-Sensitive Grammar
MCSG	Mildly Context-Sensitive Grammar
CSCG	Clone Structure Cognitive Graph
IQR	Interquartile Range.

CHAPTER 1

INTRODUCTION

Human brain has a capacity for processing information, abstract thinking, creating an abstract understanding of the environment, understanding complex concepts, and creating complex structures accompanied by a complex language that allows expressing and creation of these ideas. Humans had an interest in understanding this complex and fascinating capacity for ages starting from the oldest myths and philosophical debates to modern scientific era. Although we have a much better and objective understanding of the human mind, it still poses a significant challenge for scientists from various fields. Historically there have been attempts to understand the human mind within modern fields of study including but not limited to psychology, biology, neuroscience, philosophy, artificial intelligence and linguistics. While this challenge remains challenging to this day, much progress is being made with multidisciplinary studies under the field of cognitive science.

1.1 Language and Cognition

Linguistics, the discipline that studies the properties of languages across the world, approaches language as a multifaceted topic of study, systematically organizing its investigation into various levels to capture the complexity of human communication. These levels, or subfields, each focus on a different aspect of language structure and use. Phonology examines sound systems and patterns within languages, analyzing how phonemes are produced and perceived. Morphology focuses on the structure of words, exploring how morphemes—the smallest units of meaning—combine to form words. Syntax investigates sentence structure, studying the rules and principles that govern how words are arranged to convey meaning. Semantics investigates the meaning of words and sentences, considering how meaning is constructed and interpreted. Pragmatics extends this analysis to language use in context, examining how speakers convey and understand intended meanings in different social and communicative situations. [12]

Building upon these linguistic foundations, researchers have developed formal approaches to studying language structure and complexity. The vastness and complexity of linguistic phenomena motivated formal/mathematical approaches to language, which treat language as a structured, rule-governed system, emphasizing the abstract principles and formal rules that govern linguistic phenomena. These approaches often draw from mathematical and logical frameworks to model the underlying structure of language. A formal language is typically defined as a set of strings of symbols constrained by specific syntactic rules. Unlike natural languages, which evolve organically and are characterized by

their variability, complexity, and ambiguity, formal languages are precisely defined and unambiguous, allowing rigorous analysis and manipulation.

Formal languages have played a particularly important role in improving our understanding of the computational properties of syntax by providing a mathematical framework for analyzing and modeling linguistic structures. For instance, through the Chomsky hierarchy [13], formal languages can be classified under different levels of syntactic complexity, from regular languages that can be processed by simple finite automata to context-free languages that require more powerful computational models like pushdown automata. In other words, the hierarchy illustrates how increasing restrictions on the form of grammars reduce the complexity and expressive power of the languages they generate. This hierarchical classification helps linguists and computer scientists determine the computational resources needed to parse and generate sentences in various types of languages. By employing formal grammars, such as context-free and context-sensitive grammars, researchers aim to systematically study the rules that govern sentence structure, uncovering universal principles and language-specific variations. Moreover, formal languages facilitate the development of algorithms for natural language processing applications, such as syntax parsing, machine translation, and speech recognition, by providing clear guidelines for the computational handling of linguistic data. Finally, adding mild context sensitivity to context-free grammars allowed theoretical linguists to develop formal languages that bear stronger resemblance to natural languages in terms of their syntactic patterns. For example, mildly context-sensitive grammars, such as Tree Adjoining Grammars (TAG) and Combinatory Categorial Grammars (CCG) can capture cross-serial dependencies, unbounded dependencies, and certain types of hierarchical structures that are common in natural languages [14, 15, 16].

In addition to these formal approaches, neuroscience has provided valuable insights into the biological basis of language processing. The neuroscience of language is another related discipline focusing on the neural mechanisms underlying the acquisition, comprehension, production, and use of language [17]. Neuroimaging (e.g., PET, fMRI, fNIRS), electrophysiology (e.g. EEG, MEG), and brain stimulation methods (e.g., TMS, tCDS) are typically utilized to study brain activity and structures involved in language tasks. In addition to this, lesion studies and language disorders also play a key role in understanding the functional roles fulfilled by specific brain regions during language processing. These studies identified a vast network of cortical and subcortical regions primarily in the lateral surfaces of the prefrontal and temporal cortex, which is located primarily in the left hemisphere, with weaker activations in homological regions in the right hemisphere [18] . When aggregated across a group of participants the language networks are fairly similar across typologically diverse languages [19] and the constituent regions of the language network exhibit strong functional connectivity [20]. When individuals' brain responses are investigated individually in fMRI there can be some individual differences in the magnitude of responses observed in the language network. However, language related activations are stable within individuals over time. Finally, structured stimuli, such as sentences that convey compositional meanings, trigger a stronger response compared to unstructured stimuli, such as word lists that only express individual lexical meanings [18]. Word lists, in turn, elicit a stronger response than lists of meaningless nonwords [21, 22]. Various non-linguistic inputs and tasks generate little or no response in language areas, despite strongly activating other brain regions [23, 24].

Comparative studies that aimed to identify what is unique about human language faculty across species offer important implementation-level insights for human language faculty. These studies utilized artificial grammars to test and compare syntactic skills of different species and found that birds and monkeys can mainly deal with finite-state grammars, but not with grammars at higher levels in the Chomsky hierarchy [25, 26, 27]. Anatomical comparisons also highlighted that humans differ from

other species exhibiting syntactic skills like chimps and birds in terms of the increased size of the brain regions involved in language processing as well as the greater level of connectivity among those regions [9, 28, 29, 30]. These results highlighted, processing certain kinds of grammar requires specific neural resources, possibly linked to functionality such as a pushdown stack which is anticipated by the computational analysis of these grammar types [27].

Despite these advancements, integrating insights from linguistics and neuroscience remains a significant challenge. Studies in formal and classical linguistics operate at a functional level, whereas neuroscience of language focuses on the implementation of language faculty in the brain. Therefore, establishing relationships between some of the rules and structures identified in theories of syntax and semantics and observations from neuroscience have turned out to be challenging. Some neuroimaging studies found regions selectively sensitive to syntactic processes, dissociating them from semantic processes, but the picture turned out to be more complicated as language areas are reported to exhibit sensitivities to both syntactic and semantic information by other researchers. Therefore, some of the distinctions made at the functional level may not have a trivial neurological correspondent. This motivated the search for alternative linking hypotheses between neuroscience findings and linguistic insights [31].

1.2 Computational Approaches to Language and Cognition

Symbolic and rule-based formal languages constitute one perspective in modeling linguistic processes. Connectionism, the precursor to modern large language models, was aimed at learning linguistic rules and structures from corpus data through statistical pattern matching. Connectionist modeling problematized the existence of pre-existing rule structures by demonstrating their emergence through statistical learning. The recent increase in computational power enabled researchers to build vastly deeper networks with billions of parameters over extremely large datasets. However, neural networks, despite their success in many domains, have historically struggled with certain aspects of language processing, especially when it comes to generalizing beyond training data by learning the required rules. Large language models (LLMs) trained on vast datasets can exhibit human-level language capabilities on many language tasks. But this capability arises from statistically related tokens at a higher level. Transformers, which are the building blocks of large language models, have been shown to perform worse than LSTM models [32]. Despite their strengths in generating text in natural language, LLMs have limited performance when solving arithmetic reasoning tasks, often providing incorrect answers [33]. LLMs struggle with zero-shot arithmetic reasoning tasks, showing significant variability in performance across different datasets [34]. Connectionist neural models performed much better on processing sensory data, whereas neural networks have shown limited capacity in tasks that require algorithmic or logical reasoning compared to rule-based models. [35, 36]

Unlike rule based models, human brains are understood to have complex information processing capabilities and structures that arise from the interactions of neurons and specific brain mechanisms within biological and neurological constraints. Understanding these complex interactions raises the necessity to build models that can account for this complexity and flexibility. This study addresses this necessity by focusing on a specific problem with artificial neural network architectures as computational mechanisms with biologically realistic, sufficiently complex and trainable models. This approach allows the current developments in artificial intelligence and neural networks to be used as tools to create models of certain aspects of human intelligence. Specifically we are focused on neural networks with

augmented memory that allow us to create and test theories about how cognitive systems learn and process sequences with different neural memory mechanisms. Using this framework specifically, we try to replicate the generalization and rule learning capabilities similar to rule based systems in a computational cognitive framework that uses connectionist neural network models and memory mechanisms.

1.3 Motivation

A key motivation for this study is to understand how the cognitive systems can learn and generalize complex grammatical structures. Humans, including children, can infer intricate language rules from a relatively small number of examples and apply them to novel situations. This ability suggests sophisticated cognitive mechanisms for pattern recognition, rule extraction, and generalization. This thesis investigates the structural biases, especially the memory mechanisms that are required or beneficial for these abilities.

The ability of humans to process hierarchical structures in language, such as nested clauses, is another fascinating aspect of cognition that motivates this research. Understanding how the brain represents and manipulates these complex structures is crucial for developing a comprehensive model of language processing. By testing our models on tasks involving hierarchical structures, we hope to gain insights into the cognitive mechanisms and structural biases that enable this capability. The use of artificial grammars in this research is particularly motivated by their ability to isolate specific aspects of grammar and rule learning. Artificial grammars allow us to create controlled experiments that focus on particular computational challenges, such as hierarchical structure processing or long-distance dependencies, without other aspects of human language. Despite their limitations and omissions from capturing natural language-related phenomena, this approach still enabled us to systematically investigate some of the key cognitive mechanisms involved in grammar acquisition and generalization, providing insights that might be far more difficult to obtain when other aspects of language such as semantics and pragmatics are involved. Limiting the scope is beneficial for eliminating the complex effects that could arise from complex interaction of these phenomena and also it creates a more tractable problem that we can train models for with limited resources.

The role of memory in language processing is another key driver of this research. Working memory, in particular, has been implicated in various aspects of language comprehension and production [37]. However, the exact mechanisms by which different memory systems contribute to language processing remain unclear. Our focus on memory networks, especially those with external memory components, is motivated by the potential to model and understand these memory-language interactions more precisely. In our experiments NTMs have shown the capacity to learn the rules of various grammars where more classical recurrent neural networks like LSTMs have failed. NTMs memory mechanisms like content based and location based addressing mechanisms provide an interesting structure where we can store and recall information in various ways. These mechanisms also show similarities to structures and pathways in neuroscientific research. These similarities can allow this framework to be used for generating specific, testable hypotheses about neural activity patterns and connectivity. These hypotheses can guide neuroimaging studies, helping to interpret observed brain activity in language tasks and potentially revealing the neural correlates of specific computational processes. Moreover, the comparison between different neural network architectures in our study might suggest analogous structures or processes in the brain, hinting at possible directions for neuroscientific research. Additionally, examining these structures computationally can help us understand properties that are unique

to humans compared to those of other species. By modeling grammar and rule learning, we aim to identify the computational features that might be distinctive to human cognition, potentially shedding light on the evolutionary development of language.

1.4 Challenges and Future Directions

Despite significant advancements in the modeling of cognitive functions, several challenges remain in the field.

One of the foremost challenges is bridging the gap between formal linguistic theories and their neural implementations. Although formal linguistic theories provide a comprehensive framework for understanding language structure and processing, creating computational and neural network models that can reflect the underlying cognitive computational processes has proven difficult. The complexity of human language, with its recursive and hierarchical structures, poses a substantial challenge to creating models that can replicate these features in a biologically plausible way. This gap not only limits the explanatory power of current models but also hinders their applicability in broader cognitive and linguistic research.

Another critical challenge lies in the development of simpler and concise models that can effectively explain the specific computations leading to linguistic competence. Current models, particularly those based on deep learning, often require vast computational resources and large datasets to achieve proficiency, which raises concerns about their scalability and alignment with human cognitive processes. Human brains, in contrast, can achieve linguistic and syntactic competence with far less data, suggesting that our models may be overly complex or not fully capturing the underlying mechanisms of human language. Therefore, refining these models to be more efficient and less resource-intensive, while focusing on specific aspects such as syntax is a useful and more accesible direction for future research.

Lastly, there is a pressing need to address the differences between human language acquisition and the training processes of artificial models, particularly in terms of data requirements and computational constraints. Human language acquisition occurs naturally, with limited and often noisy input, and involves a rich context of interaction and feedback. In contrast, artificial models typically require large amounts of carefully curated data and extended training periods to reach similar levels of performance. This discrepancy highlights a fundamental challenge: developing models that not only replicate the outcomes of human language learning, but also emulate the processes involved, such as generalization from limited examples and learning in a dynamic, interactive environment also having similar structural biases indicative of simillarity at algorithmic level.

To address these challenges, researchers are exploring novel approaches such as Neural Turing Machines (NTMs) [38], which combine the learning capabilities of neural networks with structured memory mechanisms. These models offer a promising avenue for investigating the computational properties that enable human-like language processing while maintaining links to structures proposed in formal language theory.

In conclusion, the current state of research in language studies is characterized by a rich interplay between formal theories, neuroscientific findings, and computational models. While significant progress has been made in understanding various aspects of language processing and acquisition, many ques-

tions remain open, driving continued research in this fascinating and complex field, including this study.

1.5 Overview of Findings

This thesis proposes a cognitive computational framework to evaluate the syntactic learning capacity of neural network architectures when exposed to artificial grammars of varying complexities. It investigates the networks' ability to learn and generalize grammatical rules by testing them with out-of-distribution sequences that exceed the lengths of those encountered during training. The primary focus is on how different memory mechanisms embedded within these architectures influence their generalization performance. The memory mechanisms explored include augmented memory networks like Neural Turing Machines (NTMs), recurrent memory structures such as Long Short-Term Memory (LSTM) networks, and time-orthogonalizing mechanisms like Legendre Memory Units (LMUs).

As detailed in subsequent chapters, our findings demonstrate that the type of memory architecture directly impacts generalization performance. LSTM networks rely on recurrent memory, LMUs use a strategy to orthogonalize temporal history, and NTMs leverage an external memory that stores and retrieves explicit items from distinct locations. All of these architectures successfully generalized Reber grammar, a regular grammar. Both LSTM and NTM architectures generalized the A_nB_n grammar, a context-free grammar (CFG). LSTM networks succeeded in this task due to their ability to utilize a counting strategy, which is well-suited for A_nB_n , a special case of CFG that can be solved through counting. However, only NTMs, with their external memory mechanisms, were able to generalize the Mirror grammar, a CFG that requires mirroring tokens after a specific point in the sequence.

Further investigations into other grammatical structures revealed that augmented memory networks, such as NTMs, could generalize some context-sensitive grammars, but not all. Mildly context-sensitive grammars (MCSGs), a subset of context-sensitive grammars with specific structural and computational constraints, were also studied. Experiments conducted on MCSGs designed for this thesis—such as the AWA (WW^R_WW^R) and AWA-Star (WW^R_WW^R(_WW^R)*) grammars—alongside meta-analyses across various architectures [39, 40, 32, 38], suggest that the intrinsic properties of MCSGs are strong predictors of whether a grammar can be learned and generalized by memory-augmented networks like NTMs. These findings underscore the critical role of structural biases—particularly memory mechanisms—in the syntactic capabilities, rule learning, and generalization performance of neural networks.

In addition to these technical findings, this thesis also explores the relationship between memory mechanisms and syntactic processing in both artificial grammars and neuroscientific studies. There are striking parallels between the syntactic capabilities observed in humans and animals and the memory mechanisms involved in syntactic processing. This thesis aligns these parallels with neuroscientific research, suggesting that memory mechanisms, specifically the ability to store and access explicit memory, might be playing a crucial role in syntactic processing beyond regular grammars. This highlights memory as a key component of syntactic ability, particularly when tackling more complex linguistic structures.

1.6 Research Questions

This research revolves around key questions aiming to uncover the details of memory mechanisms in NTMs and their relation to human memory systems:

- **How do memory mechanisms in neural networks affect computational abilities in tasks requiring complex cognitive functions?**

  This question explores the impact of memory mechanisms of neural networks on their performance in tasks that involve syntax grammar and rule learning, with memory mechanisms like recurrent memory where memory is maintained in the recurrent state, content based memory that retrieves items based on the similarity and location based memory where memories stores to a specific location can be recalled explicitly.

- **What are the key structural biases in memory networks needed for solving problems traditionally challenging for connectionist approaches?**

  This question aims to identify and analyze crucial structural biases in memory networks that enable them to tackle complex problems beyond the reach of traditional neural networks.

- **How can the use of deep learning tools in modeling memory networks serve as a link between higher-level cognitive theories and neuroscientific understanding?**

  This question explores the potential of deep learning methodologies in connecting abstract cognitive theories with concrete neuroscientific evidence, thereby enhancing our understanding of brain functioning.

- **What role do memory mechanisms in neural networks play in computational models when compared to human memory systems like episodic and working memory?**

  This inquiry looks into the similarities and differences between memory mechanisms in NTMs and human memory systems, particularly episodic and working memory, and how these similarities and differences affect computational modeling of cognitive processes.

1.7 Significance and Limitations

This study contributes to the fields of cognitive modeling and neuroscience by enhancing our understanding of memory mechanisms within computational models. By investigating the memory mechanisms of various neural networks such as LSTMs, LMUs and NTMs, this research aims to deepens our comprehension of how memory networks can be utilized to simulate some of the high-level cognitive processes such as syntax and rule learning. This work advances the development of more refined computational models that could better mimic and understand some aspects of human cognition.

Moreover, the study aims to contrubute in bridging the gap between computational models and empirical neuroscientific findings. By integrating empirical data from neuroscience with theoretical models, it provides a framework for testing and validating hypotheses about cognitive functions and the underlying brain mechanisms. This connection between computational and biological perspectives not only aids in understanding cognitive functions such as syntax and memory but also informs the design of neural network architectures that are more aligned with certain aspects of brain's natural processes.

The implications of this research extend to both artificial intelligence and cognitive science. The findings on the capabilities of NTMs in managing complex cognitive tasks offer valuable insights for the development of AI systems with enhanced problem-solving and reasoning abilities, approaching human-like cognition. Such advancements could lead to more sophisticated AI applications that can tackle tasks requiring higher-order cognitive functions.

However, the study also has its limitations. The scope of this research is confined to certain types of neural network architectures, particularly NTMs, and their comparison with models like LSTMs and LMUs. While these comparisons are insightful, they are not exhaustive, leaving out other potentially relevant architectures. Additionally, the experiments are conducted on artificial grammars, which, while useful for understanding syntax and memory mechanisms, may not fully capture the complexities of natural language or other cognitive tasks. This research aims to provide and explaination at the algorithmic level by trying the model the underlying comptational process, the used architectures are connectionsist in nature but not directly biologically plausible. These limitations suggest that further research is needed to explore additional models and apply these findings to more diverse and naturalistic datasets.

1.8 Organization of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 presents a comprehensive review of the relevant literature and theoretical background, positioning this research within the broader context of existing knowledge. It discusses key studies, theories, and models that inform and underpin the current investigation.

Chapter 3 outlines the research methodology employed in this study. This chapter details the experimental design, data collection procedures, and analytical methods used to address the research questions. It provides a rationale for the chosen methods and discusses any relevant methodological considerations.

Chapter 4 reports the results of the study, offering a detailed analysis of the data collected. This chapter interprets the findings in relation to the research objectives, highlighting key trends, patterns, and statistical outcomes. Chapter 5 focuses specifically on the results related to mildly context-sensitive grammars. It provides an in-depth examination of the findings within this particular area, discussing their implications and how they contribute to the broader research context.

Chapter 6 presents a discussion of the results, integrating them with the existing literature and theories reviewed earlier. This chapter critically evaluates the significance of the findings, explores their implications, and discusses how they relate to other relevant work in the field.

Finally, Chapter 7 concludes the thesis by summarizing the key findings and contributions of the research. It also suggests areas for future work, proposing potential directions for further investigation based on the outcomes of this study.

CHAPTER 2

LITERATURE REVIEW

Addressing the research questions of the thesis requires a multidiciplinarry approach and a deep understanding of the theoretical and empirical frameworks that have shaped this interdisciplinary field. This chapter presents a review of the relevant literature which might be important in design of the framework and discussion of the results.

The chapter begins with an examination of cognitive computational models in section 2.1 highlighting their important role in simulating and understanding human cognition. This section discusses both symbolic and connectionist approaches, which have historically dominated the field, and evaluates their strengths and limitations in representing cognitive processes and mentions previous approaches to overcome these limitations. It discusses hybrid models that integrate elements from both approaches to overcome their individual shortcomings.

Following this, section 2.2, Deep Learning Models, which have revolutionized the field of artificial intelligence and cognitive science. This section critically assesses the successes and limitations of deep learning architectures, such as Convolutional Neural Networks (CNNs) and Transformers, particularly in the context of cognitive tasks. It also explores the importance of structural biases in these models, which enhance their ability to process sequences and manage long-range dependencies—capabilities that are crucial for modeling complex cognitive functions.

As language is a central aspect of human cognition, the next section delves into the unique capacity of humans to process Hierarchical Structures in language. This discussion compares human abilities with those of other animals, investigating the computational and neural requirements for processing such structures. The ongoing debate regarding the extent to which non-human animals can engage in hierarchical processing is also addressed, providing a broader context for the role of these structures in cognition.

Section 2.4 includes research about Neuroscience insights relevant to language and memory. It begins by exploring the Language Network in the brain, focusing on the roles of key regions such as Broca's area and their connectivity patterns. The discussion extends to the developmental and comparative perspectives, offering insights into the neural mechanisms that underpin language processing. Furthermore, this section reviews the critical Memory Mechanisms involved in language, including working memory and long-term memory, and their contributions to the storage, retrieval, and manipulation of linguistic information.

The final section, section 2.5 of the chapter provides Model Overviews of specific neurocomputational architectures that are central to this thesis: Long Short-Term Memory (LSTM) Networks, Legendre

Memory Units (LMUs), and Neural Turing Machines (NTMs). Each model is discussed in terms of its structure, functionality, and relevance to cognitive modeling, with particular attention to how these models parallel human brain functions. The discussion of NTMs includes a detailed exploration of their attention mechanisms and the combination of addressing methods, which are crucial for understanding how these models manage memory and process sequences. These models will be at the core of the proposed framework.

2.1 Cognitive Computational Models

There is a very large amount of knowledge piling up in various fields of cognitive sciences including psychology, linguistics, computer science and neuroscience. The need and benefit of building computational models are well known and there have been various attempts to model human cognition or aspects of cognition using computational tools. [41, 42]

From a computationalist point of view, we can assume that what the brain does is some kind of computation and information processing [43]. There is no reason why the computational and information processing constraints do not apply to computational processes of the brain and related models. And it would be safe to assume that this computation should be performed in a biologically plausible way that accounts for neurological and biological constraints.[43, 44, 42]

It is well known that humans have some distinct cognitive abilities that are not observed in other animals. Perhaps the most notable of them is the faculty of language [45]. This trait has been shown to be separate from the computational and externalization aspects and associated with processing hierarchical structures [46]. Humans have the unique ability to embed thoughts in other thoughts and express them. Theory of mind is considered to be a controversial topic regarding if it is unique to humans or not [47].

Classical AI and rule learning approaches have been somewhat successful at modeling higher level functions of human cognition like decision making and syntax. These models are often criticized for disregarding the computational mechanism imposed by the brain. On the other hand there have been many attempts to build computational models inspired by neuroscientific findings. Perhaps most popular being the connectionist approach. The connectionist approach in both cognitive science and machine learning are able to solve a wide variety of problems and explain many situations where the problem can somehow be defined in terms of statistical relevance [43]. So far these models have been very successful for building lower level aspects of cognition like sensory-motor processing [48, 49, 50, 51, 52].

An ideal computational model would have the ability to perform both high and low level functions of the human cognition and integrate various known facts or ideas and serve as a middle ground to account for evidence from different levels.

In scientific literature we have; neuroscientific studies about structure and functionality of the brain, behavioral data on humans and animals, cognitive science theories and studies at high level, high level conceptual information and models. And there is a gap between high level explanations and neuroscientific findings and sensory models. Most neural models and connectionist models avoid sequential and symbolic processing aspects.

The problem of closing the gap between these kinds of information and models stands as an important but a challenging one. Deep Learning has been proven itself as a powerful engineering tool that is inspired by neuroscience. Using deep learning approaches for creating falsifiable predictions and models to understand the nature of biological phenomena has also been proposed by researchers to use deep learning to study the neuroscientific aspects. [53]

Using the deep learning approach to create models, the structural bias of the neural networks has been proven to be one of the most important factors that developed the field of deep learning. Breakthroughs in computer vision were possible thanks to the structural biases provided by Convolutional Neural Networks(CNN) [49]. Transformers [49, 54] introduced an attention mechanism which provided a specific structural bias to the system in terms of how information can flow within the system, and with the introduction of this architecture has led to significant breakthroughs in natural language processing applications. Likewise Neural Turing Machines (NTM) [38] introduced a differentiable external memory access mechanism into neural networks and demonstrated that with the addition of such elements the networks can infer simple algorithms to perform associative recall, repeat/copy a given sequence, and learn to generalize n-gram patterns, which are difficult to achieve with conventional deep learning models. These kinds of problems might be easy to address in classical AI and symbol processing approaches; however, addressing these problems within the connectionist framework is important for creating models and predictions on how they can be solved in a biologically plausible way.

2.1.1 Overall Use of Cognitive Computational Models

Cognitive computational models are critical tools in cognitive science, helping researchers understand, simulate, and predict human cognitive processes. These models, which include symbolic, connectionist, and hybrid approaches, serve various functions in the study of cognition.[55]

Simulation of Cognitive Processes Cognitive computational models enable the simulation of complex cognitive processes, allowing researchers to test hypotheses about how these processes function in the brain. This simulation is essential for understanding perceptual, cognitive, and control tasks [56].

Integration of Theoretical and Empirical Research These models facilitate the integration of theoretical constructs with empirical data, ensuring that cognitive theories are both rigorous and testable. This integration helps in clarifying and formalizing theories, generating stimuli, visualizing data, and selecting models [57].

Explanation and Prediction Computational models go beyond mere description to offer explanations of cognitive phenomena. They help in understanding how and why different experimental conditions yield different cognitive outcomes [58].

Development of Cognitive Theories Cognitive computational models support the development of comprehensive cognitive theories by providing a structured way to test and refine these theories. This

methodological rigor helps distinguish between competing models and refines our understanding of cognitive processes [59].

Handling Complexity and Scalability These models address the complexity of cognitive processes and ensure that theoretical models are scalable and applicable to real-world scenarios. This capability is crucial for modeling high-level cognition and brain function [60].

Combining Multiple Levels of Analysis Cognitive computational models operate at various levels of analysis—from computational to algorithmic to implementational—providing a comprehensive framework for studying cognition. This multi-level approach helps bridge the gap between high-level cognitive functions and their neural implementations [61].

Enhancement of Cognitive Science Research The use of computational models in cognitive science helps enhance the research process by providing tools that are precise and replicable. This rigor in modeling ensures that cognitive science progresses through the development and testing of robust, falsifiable theories [62].

Overall, cognitive computational models are indispensable in cognitive science. They provide essential tools for simulating, explaining, and predicting cognitive processes, integrating theoretical and empirical research, and developing comprehensive cognitive theories.

2.1.2 Overall Use of Cognitive Computational Models

Cognitive computational models are essential tools in cognitive science that help researchers understand, simulate, and predict human cognitive processes. These models, which span symbolic, connectionist, and hybrid approaches, fulfill various roles in the study of cognition, contributing significantly to the field.

Firstly, cognitive computational models are invaluable tools for simulating cognitive processes, allowing the testing of hypotheses about functions such as perception, attention, control, learning, and reasoning[56]. This capability makes the intricate and interactive nature of these processes more tractable and testable.

Moreover, these models play a critical role in integrating theories with empirical data, ensuring that cognitive theories are both rigorous and testable. By doing so, they help clarify and formalize theories, generate stimuli, visualize data, and assist in model selection [57].

Additionally, computational models provide explanations for cognitive phenomena, giving insight into why different experimental conditions lead to varying cognitive outcomes going beyond mere description [58]. They also support the development of comprehensive cognitive theories by offering a structured approach to test and refine these theories, which is crucial for distinguishing between competing models and deepening our understanding of cognitive processes [59].

Furthermore, these models are designed to handle the complexity of cognitive processes and ensure scalability, which is vital for applying theoretical models to real-world scenarios. This ability is particularly important for modeling high-level cognition and brain function [60].

Cognitive computational models operate across multiple levels of analysis—ranging from computational to algorithmic to implementational—thereby providing a comprehensive framework for studying cognition. This multi-level approach bridges the gap between high-level cognitive functions and their neural implementations [61].

The use of computational models enhances cognitive science research by providing precise and replicable tools. This methodological rigor promotes the advancement of the field through the development and testing of robust, falsifiable theories [62].

While cognitive computational models provide valuable insights into the mechanisms underlying cognitive processes, several limitations must be acknowledged. First, these models often necessitate the simplification of complex phenomena. Due to the intricate nature of cognitive functions, models tend to focus on key aspects while treating other complexities as noise. This simplification, while necessary for tractability, can result in a loss of nuance and may fail to capture the full complexity of cognitive processes [63]. Moreover, trained cognitive computational models often suffer from limited generalizability. Models developed under specific conditions or with particular datasets may not generalize well to diverse populations or real-world scenarios [64].

Another drawback is the lack of biological plausibility for many cognitive computational models. They can often fail to accurately reflect the underlying neural mechanisms. As a result, there might be a disconnect between the model's functional predictions and the biological realities of the brain [65].

Also, the computational demands of some sophisticated cognitive models can be substantial. High computational costs can limit the accessibility and practical application of these models, particularly in contexts where computational resources are constrained [66].

2.1.3 Connectionist Models and Symbolic Models

Table 1 offers a comparison between connectionist and symbolic approaches to cognitive modeling. This table is adapted from Morris (1989) [67] where some information is added by the author to elaborate on some of the points further based on recent developments in ML/AI and Cognitive Science.

Connectionist and Symbolic Models seem to explain different aspects of human cognition better and work well with different sets of problems.Connectionist systems seem to be built in a way that captures intrinsic properties of the biological substrate and seems to be closer to how biological processes operate. On the other hand, symbolic and rule-based systems offer models that can model higher-level human cognitive processes like problem-solving and language.

Table 1: Table comparison of Connectionist and Symbolic Models

	Connectionist Models	**Symbolic Models**
Knowledge Storage	Connection Strengths	Static copy of a pattern
	Does not use external memory	Uses external memory
Retrieval	Recreate (a similar) pattern	Map with LTM
Knowledge and	Intimately Related	Related but separable
Processing	(Usually locally Processed in parallel)	Flexible processing but need to transfer information
Learning	Adjusting connections or their strength	Better Rule formations
	Solves some problems good like statistical relations	Some kind of learning are not well represented as rules (conditioning)
	Harder to learn complex non-local patterns	Complex patterns can sometimes be better represented as a set of rules (algorithms)
	Cannot generalize non-local patterns	
Representations	Usually distributed	Local, Symbolic
	Flexible and gradual representations	Information needs to be moved around continuously
	Can better represent analog variables and conditions	Better explains higher abstractions (like verbal representations)
	Embedded in activations cannot be easily transferred this makes modelling modular processing harder	Represents categorical information better
Processing	Parallel	Local, Symbolic
	Happens nearly instantly	Better for processing sequential information
	Computationally better for processing local information	
Control	No specific control mechanism required	Order and type of instructions and rules need to be controlled
	Regulates itself	Requires attention mechanisms
	No attention mechanism needed	

A very popular rule-based learning system is the production system which has the following properties [68]:

- A production rule base that stores the rules
- A working memory to store initial, intermediate and final results
- A control structure for coordinating the running system

Usually, rule-based systems are criticized because they do not always have an explanation for where the initial rules may come from; they cannot deal very well with noisy inputs. Also, they are not implemented in a way that has graceful degradation. Also, it may be harder to map these systems to biological substrates [68].

Here, avoiding the dichotomy between these two types of systems and trying to build a system that can offer explanations from each side can offer some benefits [51, 69]. There are a variety of hybrid models that try to combine principles of these two schools.

- CONSYDERR: a two-level hybrid architecture for structuring knowledge for common sense reasoning [69]
- SCREEN: Spoken language understanding model [70]
- ACT-R: Rule-based cognitive architecture that uses connectionism inspired memory elements [71]
- CLARION: A cognitive architecture that makes a distinction between implicit and explicit processes that uses both rule-based and connectionist components [72].

Unlike classical connectionist models hybrid systems usually operate by having connectionist components in an architecture like symbolic systems and the control is usually provided externally.

There are various forms of hybrid neural architectures [73, 74]:

- **Unified Neural Architectures** use only connectionist representations, and the nodes usually have conceptually symbolic meanings (e.g. a node for each word)
- **Transformation Architectures** use neural networks to extract or inject symbolic rules. They can turn symbolic representations into neural representations and vice versa.
- **Hybrid Modular Architectures** use both symbolic and neural modules to work together. This property provides the ability to have modules that can solve problems with the best-suited modeling approach. Usually, neural representations are preferred more for sensory processing and symbolic representations are preferred for top-down processing.[74]

With the recent developments in neural networks, it is possible to have the right structural biases to build a completely differentiable neural network that can provide symbolic-like processing This opens up the possibility of having modules that can use distributed neural representations and still try to solve symbolic processing problems.[68, 38, 75]

Connectionist approaches have a track record of success in some aspects of cognition but they are still lacking at others. Considering the powerful modeling and information processing tools they are, they should be improved in the ways that symbol processing models are good at. This might require re-evaluating the existing structures and building new structures in connectionist models.

2.1.3.1 Benefits of Connectionist and Symbolic Computational Models

Two broad categories of computational cognitive models are symbolic and connectionist models, each offering unique advantages in understanding cognitive processes.

Benefits of Symbolic Models

1. **Clear Structure and Interpretability**: Symbolic models use structured, rule-based representations of knowledge, making them highly interpretable and easy to understand. This clarity aids in mapping cognitive processes like logical reasoning and language processing. [76].

2. **Efficiency in Rule-based Inference**: These models excel at tasks requiring strict adherence to rules and logic, such as mathematical reasoning and complex problem-solving [51].

Benefits of Connectionist Models

1. **Biological Plausibility**: Connectionist models, inspired by neural networks, offer a biologically plausible approach to understanding cognition. They simulate networks of neuron-like units, capturing aspects of human neural processing [77].

2. **Handling of Ambiguity and Noise**: These models excel in dealing with ambiguous, noisy, and incomplete data, much like human cognition does. They can adaptively learn from experiences, improving their performance over time [78].

3. **Parallel Processing**: Connectionist models leverage massively parallel processing, making them well-suited for tasks that require simultaneous processing of multiple pieces of information, such as visual perception and pattern recognition [79].

4. **Learning and Adaptation**: These models can learn and adapt through processes like backpropagation, making them capable of improving their performance based on feedback, much like human learning [80].

Benefits of Hybrid Models Hybrid models, which integrate symbolic and connectionist approaches, combine the interpretability and structured reasoning of symbolic models with the adaptability and robustness of connectionist models. This synthesis can lead to a more comprehensive understanding of complex cognitive tasks, such as language processing and cognitive development [81].

Both symbolic and connectionist models have distinct benefits that contribute to the field of cognitive science. Symbolic models provide clear, interpretable frameworks for rule-based reasoning, while connectionist models offer biologically more plausible, adaptive approaches to learning and pattern recognition. Hybrid models effectively combine these strengths, advancing our understanding of higher-level cognitive processes.

2.2 Deep Learning Models

In recent years deep learning methods which are large neural networks have been successful in a variety of problems. There are several reasons behind this.

One important factor is the increase in computing power and the usage of GPU's. With the increase of computational power, the motivation to create larger datasets that can be used for training increased.

The quantity and size of the datasets were an important factor in the success of deep learning methods. By better understanding the search and problem space, advanced optimization algorithms could be developed. Also the development of better and more efficient architectures that provide structural biases suitable to the nature of the problem was a major factor in the success of deep learning. [48]

2.2.1 Where Deep Learning Succeeds and Fails

After the success of vision models, there has been a significant amount of interest in deep learning models that provided very successful models on object classification [49], object detection[52], playing go [82] that can reach and even surpass human-level capabilities. However, with the increasing size of the networks, the training data was not very helpful for every problem. Even in tasks that include well-defined problems like graph algorithms, neural networks still cannot perform as well as algorithms. Moreover, generalizing the learned patterns to perform on novel inputs that were not included in the training is another aspect of the deep learning methods that need to be improved. Also, we cannot currently build models that can transfer knowledge from one domain to another flexibly.

Natural language processing is a major field that did not share the success of vision in deep learning models until the transformer architecture was used with very large networks. Considering the machine learning aspect, there may be a couple of reasons: Firstly, having a good numeric representation is one of the most important factors in machine learning problems. Since images can be represented in a space of different intensities and using different channels, digital image representations are already close to the nature of light and also how biological systems sense them on a sensory level. Unlike images, language is usually stored as digital text composed of characters. Nature of these characters or words are symbolic in nature where the used tokens are almost arbitrary. Also the actual meaning of a word is often strongly dependent on the context that it is in. We can also have higher level representations, but at the input level we can say that digital images provided a better representation. Also in vision, important low level features are usually grouped together whereas in language, local changes can have a very significant effect on structure and meaning. How a sentence starts might affect how it ends, or a single word after an utterance can completely change the meaning. Another factor is the scalability of the image representations. We can have a similar image by selecting a fraction of the pixels but in language removing these representations can completely take away the meaning.

However, in recent years, there have been important developments in the NLP scene. With the help of the transformer models [54], scalable models could be built for NLP problems. Regardless of the impressive results, they work by having all the sequence available as an input and do not process it sequentially. This makes it difficult to scale with the sequence size because of the need to increase the input size of the models. Moreover, such models require huge amounts of data. It is also shown that self-attention is not suitable for modeling periodic regular languages or basic recursion [83].

So far with the addition of correct mechanisms, deep neural networks have achieved substantial improvements, adding correct structural biases seems to improve the performance and potential for the neural networks drastically. However, for many problems, the structural biases imposed by popular architectures still fails to replicate some aspects of human language and cognition.

2.2.2 Structural Biases at Neural Networks

Providing the right architectural biases enabled deep learning methods to have success in a variety of problems. After Convolutional Neural Networks(CNN) were introduced those kinds of networks achieved dramatic performance increase for vision tasks. [49] Theoretically, it is possible to construct a fully connected neural network equivalent to a CNN. The inclusion of convolutions provided a bias to increase the importance of relations of pixels that are located close to each other. This enabled the system to create a filter-like structure and build more meaningful representations as the layers were added on top of each other [84].

Classical RNN's were able to process tree-like structures, but they could not deal well with longer dependencies because the information from previous inputs degrades over time. Therefore the inclusion of the right kind of architectural bias improved sequence processing dramatically.Long Short-Term Memory (LSTM) networks [85] and other similar structures have the ability to choose whether to remember or forget the information from the current time step. This bias added the ability to choose how the information will be stored depending on the content of the information.

Similarly, the transformer architecture introduced a better structural bias with self attention mechanism.by processing the text in a non-sequential way and having a self attention mechanism between tokens, it was able to provide a better solution for evaluating each token based on the context of other tokens [54]. The parallel nature of transformers also provides advantages for training on large infrastructure, but makes it computationally more expensive and requires much more memory. This is different from the sequential and time-based nature of how the human brain handles language. [**?**]

Another method that was used by deep learning methods and provided successful results was the attention mechanisms which can provide memory access mechanisms that can be trained. This can help remove the information bottleneck that is present in other kinds of models by enabling access to previous inputs [86] or learned representations [38]. These attention mechanisms added the right kind of structural bias to the model to cope with the longer dependencies and complex patterns [38, 87]. For example only after the inclusion of these mechanisms that these models were able to play Atari games [88] and beat human experts in games such as Go [88, 82, 89].

2.2.2.1 Attention Mechanisms

Attention mechanisms are useful at providing information about either inputs other than the current input or a learned representation from other inputs. They are basically memory access mechanisms that can be defined as key-value maps of a query to some contents in the memory. It should be noted that the "attention" mechanism in the machine learning domain is not the same concept as the concept of attention used in cognitive science in general. However, there may also be some similarities at a functional level between the use of attention across both domains. For instance, human working memory can be associated with attention and in some cases interpreted as "internal attention" [90]. It would be debatable to suggest if attention in machine learning models maps well onto attention in cognitive science, but it would be safe to consider the attention mechanisms in machine learning as a candidate for **memory control structures**.

Attention mechanisms work by running a query to find similarity between the keys and the query and then by calculating a weighted sum of the values according to the similarity.

$attention(k, v, q) = (weight(k_i, q).v_i)$

and

$weight(k_i, q) = softmax(similarity(k_i, q))$

Similarity function here can be additive similarity which is a single layer neural network that calculates a weighted sum following a $tanh$ activation function, multiplicative similarity which is the dot product of keys and query. If k and q are normalized it works as cosine similarity. Table 3 summarizes the different types of attention models used in machine learning.

Table 2: Comparison of attention mechanisms in various neural networks

	Keys	**Values**	**Queries**
Recurrent	Input	Input	Hidden State
Self-attention	Input	Input	Current Input
Learned self-attention	Input	Input	Learned
Memory Network	Learned	Learned	Current Input

Recurrent attention uses the hidden state to query the inputs and calculates the weighted sum of input. While self attention uses the current input to query other inputs and calculates a weighted sum of inputs. Learned self-attention uses a learned representation during training to query the inputs and calculate a weighted sum of the inputs. And memory networks act as an umbrella term for models that work with a working memory during inference or training. They usually learn the key and the value pairs which constitutes the memory, and use the current input to query to get some kind of weighted sum.

Neural Turing Machines [38] are good examples of memory networks. NTMs operate similarly to Turing machines. They have access to an addressable memory. Since they are completely differentiable, it is possible to train them by gradient descent. They can learn to access the memory and store information. All of these operations are also differentiable so NTMs can learn:

- when to read information from memory
- when to store information to memory
- where to read information from memory
- where to store information to memory
- process information according to input and contents

Differentiable Neural Computers [75] are also memory networks that represent an improvement over Neural Turing Machines. They are similar to NTMs with notable differences. One is a link matrix that keeps track of recently accessed memory locations which allows it to retrace its steps. Another improvement is an extra memory management mechanism that is able to free up memory and allocate it, using gates to control how much memory to free and to rank the most used data. These changes are engineered mechanisms to increase performance and yield more successful results.

2.3 Language

What separates humans from other animals in terms of cognition is one of the widely asked questions in cognitive science. Language is one of the most popular (if not the most) faculty specific to human cognition.

Chomsky theorized that children's ability to learn language cannot be explained in terms of just statistical relations of words. He argued that human languages all have common properties and all children learn language at about the same period. Also, with the poverty of input argument, he states that there are not enough variety of inputs for the children to learn the grammar from scratch. He theorized language is governed by large number of innate rules and principles. The Universal Grammar hypothesis suggests that there are, unconscious, innate set of constraints that define if a sentence is correctly formed [91, 92]. Language is a trait that every healthy human child can learn with limited amounts of input and supervision.

There is also evidence supporting that the language trait is independent of speech and any form of communication. While language can be externalized by speech, signing, or in written form, it plays a crucial role in human cognition beyond just communication. Although it can be externalized by various different sensory-motor interfaces, there seems to be an internal independent operation mechanism of language that is dissociated from externalization of language. [46, 25]

Friederici & Chomsky et al. states that "core human language faculty consists of a biologically determined computational cognitive mechanism that recursively assembles a potentially infinite array of hierarchically structured expressions, an ability apparently unique to humans" [46]

2.3.1 Hierarchical Structures

Some animals and humans can create and process linear structures that have linear dependencies. Studies with sign language and artificial language shows that ability for open ended combinatory manipulation or understanding hierarchical structures are unique to humans. [93, 25, 1]

At the core of the language faculty lies the syntax. Only humans have the capacity to embed phrases within phrases. This has been demonstrated by linguistic properties of human languages and artificial grammars. [46, 91, 25]

Previous work on syntax in animals showed that animals could recognize regular grammar patterns but not context-free and beyond [94]. Birds such as pigeons and songbirds can learn to recognize and categorize sequences based on simple probabilistic rules. They can generalize and categorize vocal strings based on phonetic features and learn simple patterns like reduplications. [30, 26]

Hierarchical structures are defined by supra-regular grammars (at least context-free grammars). Computationally we need push-down automata (PDA) or more complex mechanisms to process such grammars and these mechanisms require a stack or equivalent memory mechanisms.

The research on whether or not animals other than humans can process hierarchical structures is an ongoing debate in the literature. Some of the important findings are as follows.

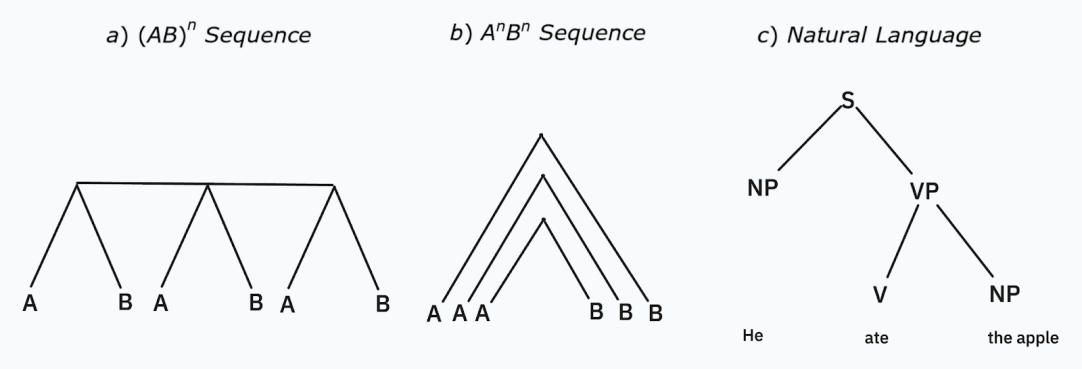


Figure 1: "Artificial strings and natural grammars. (a) Strings of the format $(AB)^n$, in which each A-category item is followed by a B-category item. (b) Consecutive sequences of equal numbers of A-category items followed by B-category items can be recognized without necessarily building hierarchical structure, by simply verifying that the number of A-category members to the left match the number of B-category members to the right. Such sequences can also be learnt by songbirds. (c) By contrast, natural language structures are always hierarchical and must be processed as such." [1]

Birds and Simple Rules Birds such as pigeons and songbirds can learn to recognize and categorize sequences based on simple probabilistic rules. They can generalize and categorize vocal strings based on phonetic features and learn simple patterns [95].

Macaques and Simple Rules Rhesus macaques have shown the ability to learn and generalize the statistical structure of sequences generated by simple artificial grammars, such as those involving adjacent relationships [96].

Humans vs. Monkeys Humans can learn both adjacent (local) and non-adjacent (long-distance) relationships in sequences, showing sensitivity to complex hierarchical structures. In contrast, macaques showed sensitivity primarily to adjacent relationships, struggling with non-adjacent dependencies [97].

Implicit Learning of Recursive Context-Free Grammars Humans can implicitly learn context-free grammars that involve hierarchical organization and long-distance dependencies. Studies have found that participants acquired unconscious knowledge of grammatical classes, suggesting that context-free grammar learning can occur without explicit awareness [98].

Baboons and Context-Free Grammars Research based on response time irregularities suggests that baboons could learn to process sequences following a context-free grammar with a mirror structure but struggled with context-sensitive grammars (repeat grammar) requiring more computational power. Closer examination of the experiments reveals a sensitivity with shorter sequences, but response differences begin to diminish when the target length reaches 6, which might indicate a sensitivity towards context-free grammars but a memory limitation [94].

Criticism Against Baboon Research Poletiek et al. argue that the suggested baboons' ability to process center-embedded structures could indicate similar origins for such structures in human language. However, they contend that the baboons' behavior likely stems from simple associative learning rather than an understanding of complex grammatical rules. The researchers emphasize the need for caution when interpreting animal behavior as evidence for human linguistic capabilities [99].

Context-Free vs. Regular Grammars Theoretical work has explored the conditions under which context-free languages can be approximated by regular grammars. These studies help understand the boundaries between the capabilities of different species in processing linguistic structures [100].

Human Children and Macaque Monkey Differences Jiang et al. (2018) demonstrate that macaque monkeys can be trained to produce complex spatial sequences that require supra-regular grammars, a capacity previously thought unique to humans. Monkeys learned to generate and generalize time-symmetrical embedded sequences, although they did so more slowly and less efficiently than preschool children, who used a chunking strategy. This suggests that while nonhuman primates can learn complex grammatical structures, humans may have a unique advantage in the speed and strategy of learning. A notable difference between human children and macaque monkeys was the length of the sequences, where monkeys could initially perform up to sequence lengths of 2 and 5, and after extensive training, this could be raised to 4 and 5. Human children could learn much faster and up to a higher sequence length, which might suggest a memory-related difference [101].

Challenges in Experimental Design Many studies investigating animals' ability to learn CFGs suffer from issues related to the stimuli used, controls in the test phase, and the clarity of the tasks presented to the animals. These problems create ambiguity about whether animals are learning the grammar or simply responding to specific cues [102].

Difficulties with Animal Studies It should be noted that artificial grammar studies are especially difficult in animal research. Because there is no direct way to explain the task and expect a response, researchers must rely on indirect methods to assess an animal's ability to recognize and generalize grammatical patterns. This often involves extensive training periods and carefully designed experiments that can differentiate between true pattern recognition and simpler associative learning. One common approach is to use operant conditioning techniques, where animals are rewarded for responding correctly to stimuli that follow the artificial grammar rules.[103] However, this method presents its own challenges, as it can be difficult to determine whether the animal is truly grasping the underlying grammatical structure or simply memorizing specific sequences. Another method is measuring response time, which can provide insights into the cognitive processing of grammatical information. Response time measurements are based on the assumption that animals will respond more quickly to familiar or expected stimuli that follow the learned grammatical rules, and more slowly to violations or unexpected sequences.[94] This technique can be particularly useful in revealing subtle differences in how animals process grammatical versus ungrammatical sequences, even when their overt behavioral responses might be similar. But this method also suffers from problems of individual variability, motivational factors like fatigue or distraction, habituation effects where response time is decreased with training regardless of stimuli and being a secondary effect rather than the direct behavioural output which might be influenced by other factors.

Overall, the research suggests significant differences in humans and other animals' ability for syntax. But the argument for taking context-freenes as a distinctive point is up for debate. For the research claiming that some animals like macaque monkeys or baboons are sensitive to some context-free grammars, there is a possibility that simpler strategies, familiarity with input sequences and the experiment design might be a contributing factor. Even with the current state of research a common pattern is, in the cases animals show sensitivity to context-free structures this capacity diminishes quickly with the sequence length. Their inability to show sensitivity at longer sequence lengths as much as 6 items or higher, might suggest relation to memory mechanisms or strategies.

Also there is some research claiming that, to be able to process representations in a hierarchical way it is hypothesised that a merge-like operation is needed and with that kind of operation structures in human languages can be represented [104, 46]. This can be an important factor for neuro-symbolic capacities and handling more complex grammars flexibly.

2.4 Neuroscience

2.4.1 The Language Network

Neuroscience of language is another related discipline in the study of language, focusing on the neural mechanisms underlying the acquisition, comprehension, production, and use of language [17]. Neuroimaging (e.g., PET, fMRI, fNIRS), electrophysiology (e.g. EEG, MEG), and brain stimulation methods (e.g., TMS, tCDS) are typically utilized to study brain activity and structures involved in language tasks. In addition to this, lesion studies and language disorders also play a key role in understanding the functional roles fulfilled by specific brain regions during language processing. These studies identified a vast network of cortical and subcortical regions (Figure 2), primarily in the lateral surfaces of the prefrontal and temporal cortex, which is mainly localized in the left hemisphere, with weaker activations in the homological regions in the right hemisphere [2].

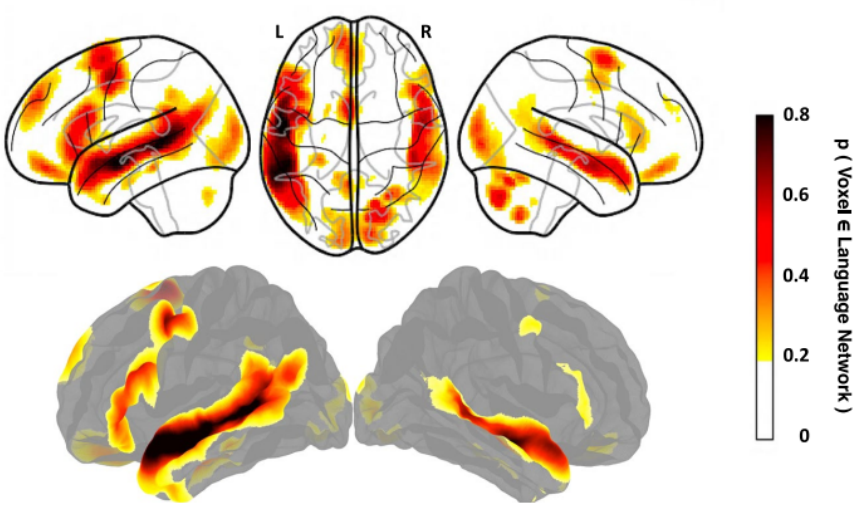


Figure 2: Brain regions implicated in the language network based on >800 individuals from (Lipkin et al., 2022, p.2) [2]

When aggregated across a group of participants the language networks are fairly similar across typologically diverse languages [19] and the constituent regions of the language network exhibit strong functional connectivity [105] . When brain responses are investigated individually in fMRI, there can be some individual differences in the magnitude of responses observed in the language network [21]. However, language related activations are stable within individuals over time [106] . Finally, structured stimuli, like sentences conveying compositional meanings, trigger a stronger response compared to unstructured stimuli, such as word lists that only express individual lexical meanings. Word lists, in turn, elicit a stronger response than lists of meaningless non-words [21, 22]. Various non-linguistic inputs and tasks generate little or no response in language areas, despite strongly activating other brain regions [23, 107, 24]. This suggests that the language network exhibits specialization towards processing linguistic content. However, associating specific regions and networks functional roles have turned out to be controversial and an active area of investigation within cognitive neuroscience of language [17]. The classical model for the neurobiology of language is based on language deficiencies reported by Broca and Wernicke of patients who suffered from distinctive damage to specific regions in their left hemisphere. In this model, the infamous Broca's area, which is located in the left inferior frontal gyrus, is associated with language production, whereas the Wernicke's region in the left superior temporal gyrus is associated with language understanding (Figure 3). These regions are connected to each other via the arcuate fasciculus, enabling the coordinated use of these faculties during language production and understanding. The model is currently considered obsolete due to challenges in accounting for various empirical phenomena obtained via imaging studies of healthy language speakers and patients suffering from different kinds of language atrophies [3, 108]. However, these regions are still considered vital parts of the language network, whose specific functional roles are continuously refined by studies.

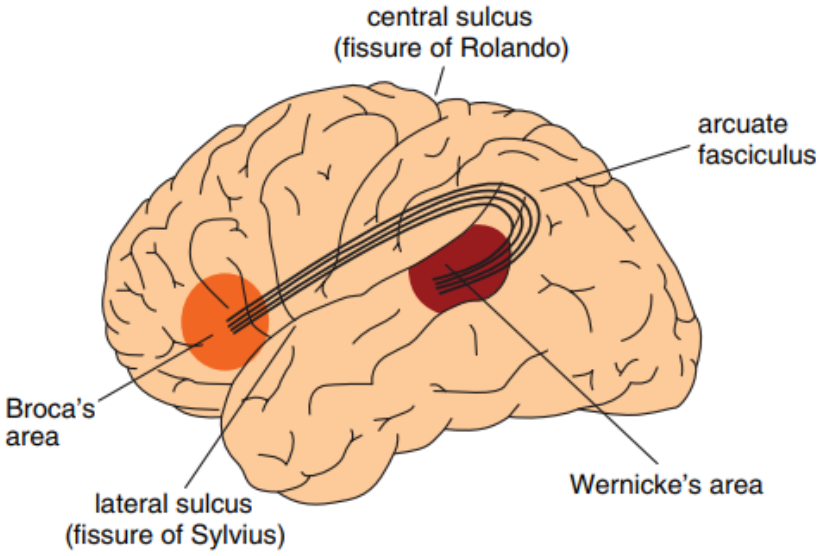


Figure 3: The classical model of the neurobiology of language from (Hagoort, 2014, p. 137). [3]

2.4.2 Functional Role of the Broca's Area

As a key component of the classical model, Broca's area has received considerable attention in neuroscience of language research [109]. Earlier studies aimed to characterize the functional specialization of this region. A recurrent theme in these studies highlight the role of this region in syntactic processing of sentences" [110]. Moreover, regions within the inferior frontal gyrus encompassing this region are also associated with verbal working memory and the phonological loop [110, 111]

Aphasia studies suggest that damage to this brain area is characterized by losing the ability to produce language whether it is spoken, signed or written form [112].This aphasia impairs syntax for both forms. Patients also have problems with comprehension related to syntax [113]. A typical patient with Broca's aphasia will misinterpret "the man is bitten by the dog" by switching the subject and object to "the dog is bitten by the man." [112]. Such observations led to the functional interpretation of this region in relation to speech production in the classical model.

Related imaging studies on this region offer evidence that Broca's area, particularly its sub-area BA 44, plays a crucial role in syntactic processing by executing the "merge" operation, which combines words into hierarchical structures [114]. This process is fundamental to understanding and producing grammatically correct sentences.

Further evidence of cortical tracking of phrasal and sentential structures was provided by a MEG study by Ding et al.[4], where participants listened to speech segments including syllable, phrase and sentence level structures. The fundamental frequencies of these structures matched with the peak frequencies observed in the neural oscillations over the language network (Figure 4). This suggests that "...during listening to connected speech, cortical activity of different timescales concurrently tracked the time course of abstract linguistic structures at different hierarchical levels, such as words, phrases and sentences" [4]. Sentential and phrasal rate responses were stronger in IFG (Broca's region), supe-

rior temporal gyrus (STG) and temporoparietal junction (TPJ). In contrast, the syllabic-rate response to sentences was stronger in broad cortical areas, including the temporal and frontal lobes, indicating processing of abstract linguistic structures at multiple scales.

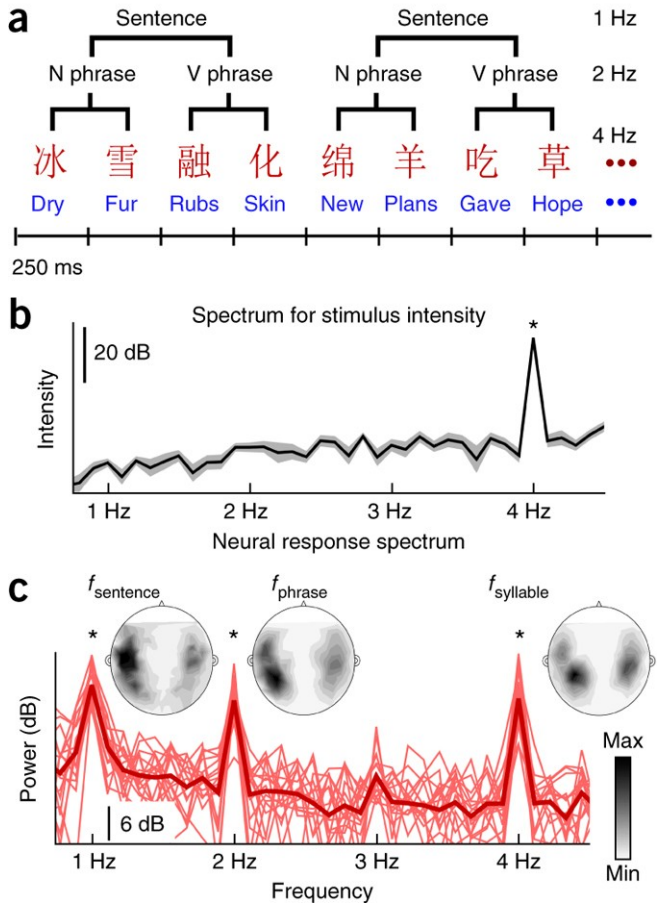


Figure 4: Frequency distribution of neural oscillations match the frequency of syllabic, phrasal and sententiaı structures in speech from (Ding et al., 2015, p. 159). [4]

As suggested by the distributed cortical regions tracking multilevel linguistic structures, it's important to note that the function of Broca's area extends beyond just syntax, and syntactic processing does not solely take place in this area. While related studies acknowledge the traditional view of Broca's area being associated with language production, they suggest that it participates in various linguistic processes, including phonology, articulation, syntactic structure building, and semantic operations [3, 115].

The recent literature in neurolinguistics highlights that Broca's area doesn't function in isolation but dynamically interacts with other brain regions to process language. This network perspective suggests that the contribution of Broca's area to language is shaped by its connections with other language-related regions, particularly in the temporal and parietal lobes [116, 117, 3]. For example, during syntactic processing, Broca's area shows stronger connectivity with posterior temporal regions, while tasks requiring cognitive control lead to increased interaction with other frontal regions [118]. BA 44

interacts with regions like the posterior superior temporal sulcus/gyrus (pSTS/STG) during language comprehension. The pSTS/STG integrates syntactic information from BA 44 with other linguistic information, such as thematic roles and semantic meaning, for comprehensive sentence comprehension [117, 114]. One prominent theory proposes a central role for Broca's area in unifying different linguistic information types [3]. This might involve integrating syntactic structures with semantic information or combining words based on their syntactic roles [119]. This suggests a more complex and dynamic role for Broca's area than previously thought. Further supporting the complex role of Broca's area, research has revealed that its activity and connectivity levels fluctuate depending on the task and processing stage [118]. For example, a study by Kunert, Hagoort et al. [120] showed an interaction between music and language syntax in Broca's area, suggesting that this region might be involved in integrating different types of structured information. It is also notable that this brain region is also used for tasks other than language including musical harmony [121]. Research shows BA 44 is involved in processing hierarchical structures in other domains, such as music, suggesting a more general neural mechanism for processing structured information [120]. However, a common pattern seems to be that it is involved in the processing of hierarchical and tree-like structures.

In summary, while a distributed network supports language processing, Broca's area, specifically BA 44, plays a central role in constructing hierarchical syntactic representations. This region's specific activation patterns, its connections to integrative processors like the pSTS/STG, and its involvement in other hierarchical processing tasks highlight its crucial role in building, manipulating, and temporally holding syntactic structures during language processing. Details on how Broca's area is involved in syntactic processing are still discussed. There seems to be evidence supporting both the use of working memory and direct role in processing.

2.4.3 Structural and Functional Connectivity Analysis of Language Related Regions

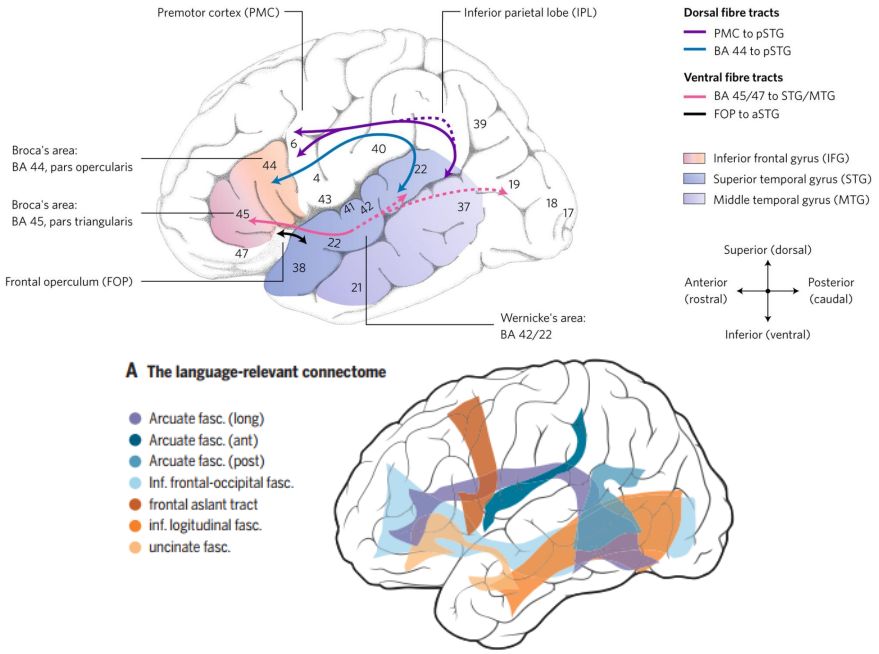


Figure 5: Summary of connectivity patterns (top) among language regions (Friederici, 2012, p. 263) [5], fiber tracts (bottom) structurally connecting the language related brain regions (Hagoort, 2019, p. 2) [6]

The increasing prominence of the network view motivated studies focusing on the structural, functional and effective connectivity relations within the language network. Diffusion tensor imaging and fiber tractography studies as well as functional connectivity analysis of fMRI signals obtained during language tasks identified dorsal and ventral pathways in the human brain that connect the temporal and frontal regions in the left hemisphere (Figure 5). This picture also positions Broca's region and its vicinity in the IFG as a hub of multiple pathways within the language network.

Imaging studies focusing on syntactic processes further refined the role of regions like BA44/45 within the language network. The dorsal pathway (blue and purple lines in Figure 5) is crucial for processing syntactically complex sentences. The connection strength and myelination of this track is directly related to syntactic abilities, especially in the developing brain [117]. The ventral pathway is another important connection from superior temporal gyrus to Brodmann Area 45 and is strongly associated with carrying semantic information. [46]

Of particular relevance to the current thesis, a study by Friederici et al. [7] investigated the activation and connectivity patterns elicited by two different types of grammar, namely a finite state grammar $(AB)_n$ and a phrase structure grammar $\mathrm{A}_n\mathrm{B}_n$, and found significant differences. In the first scenario where the dependencies are linear and there are no hierarchical structures, the main activity was seen in the Frontal Operculum (Figure 6). Moreover, the dorsal pathway associated with syntax is not activated. In the second scenario where there is a hierarchy in the input data, there is a strong activation

in the Broca's area (BA 44) along with the frontal operculum. Also the dorsal pathway carrying information to that area is strongly activated as well along with the ventral connections [7].

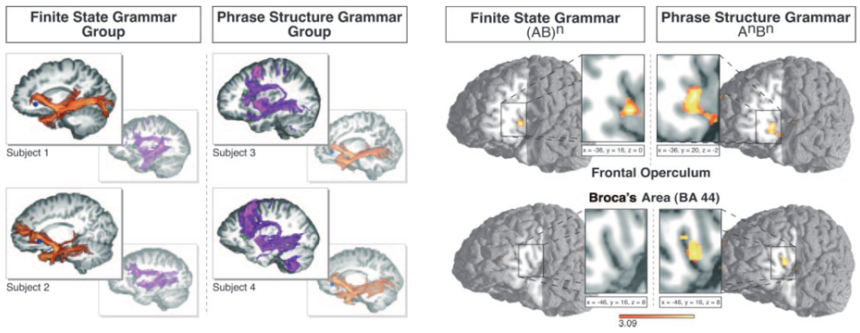


Figure 6: Different activations and connectivity patterns elicited by grammars of different complexity (Friederici et al., 2006, p. 3-4) [7]

2.4.4 Developmental and Comparative Perspectives

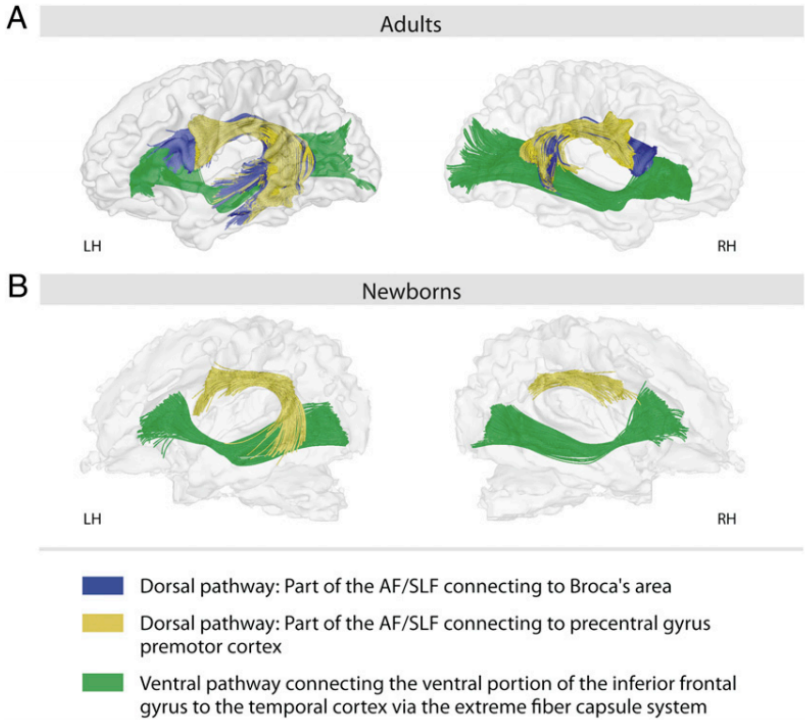


Figure 7: Pathways within the language networks of adults and newborns (Perani 2011, p. 3) [8]

Developmental changes in the human language network and comparisons of neuroanatomical properties across different species provide further insights into the role of specific structures in the language

network. Data acquired from developmental brain studies largely align with structural and imaging properties of the adult language network. However, there are some key differences, such as the lack of a fully connected dorsal pathway in newborns. Newborns also lack syntactic abilities to process hierarchical structures at that age. Myelination of the dorsal tract is completed around 1.5 years (blue section in Figure 7), which aligns with the time when infants begin to develop syntactic abilities [46, 8].

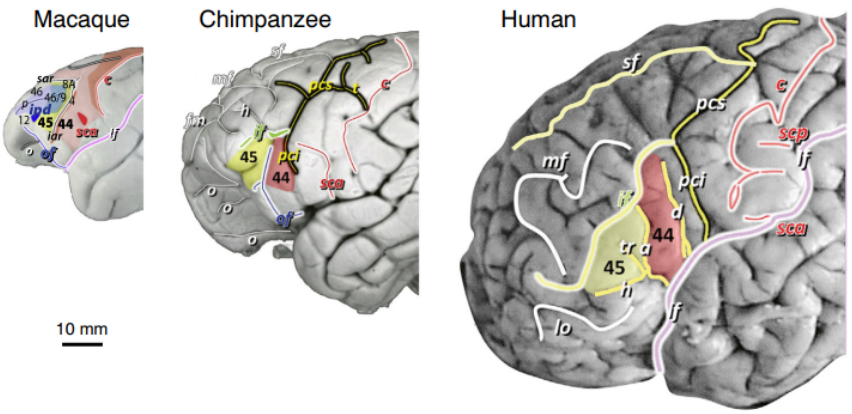


Figure 8: Anatomical comparison of BA 44/45 regions across macaque monkeys, chimpanzees, and humans (Friederici, 2023, p. 793) [9]

Anatomical comparison of gray matter volume between macaque monkeys, chimps and humans suggest that the human prefrontal cortex including BA 44/45 regions is 1.9 and 1.2 times larger in humans than macaques and chimps, respectively [28]. A cross-species cytoarchitectonic comparison revealed that Broca's area in the left hemisphere, but not in the right hemisphere, expanded in relative size during evolution from the chimpanzee to the human brain [29].

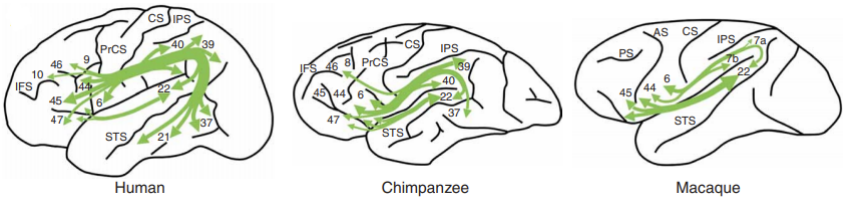


Figure 9: A comparison of structural connections among the language related regions in the human, chimpanzee and macaque brains (Rilling et al., 2008, p. 427). [10]

Cross Species studies show that the dorsal pathway is not well developed in species other than humans. Even the closest animals do not have strong connections to analogous of BA 44 in those animals [10] . Also from the 3d model of the macaque brain below it can be seen that there is a gap between the dorsal and ventral fibers (i.e. arcuate fasciculus), resulting in an incomplete fronto-temporal loop. These results also align well with behavioral data regarding the syntax capabilities of those animals.

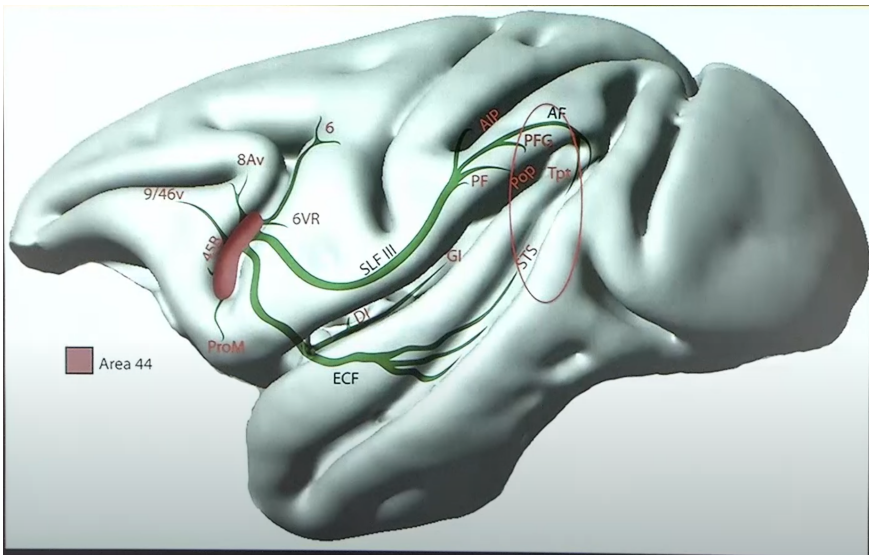


Figure 10: Anatomical connections within the left hemisphere in the macaque brain (Frey, Mackey, and Petrides 2014, p 19.) [11]

2.4.5 Memory Mechanisms and Language

Memory mechanisms such as working memory and long-term memory play a critical role in language processing and comprehension. These mechanisms involve various cognitive and neural processes that support the storage, retrieval, and manipulation of linguistic information.

2.4.5.1 Working memory

Working memory is crucial for temporary storage and manipulation of information during language processing. It includes components like the phonological loop (for verbal and acoustic information), the visuospatial sketchpad (for visual information), the central executive (an attentional control system), and the episodic buffer [122]. Verbal working memory (WM) is closely linked to language production processes. The architecture for verbal WM involves domain-specific mechanisms for serial ordering, which are crucial for language production and comprehension. Positional, lexical, and phonological similarity constraints in WM are paralleled in computational models of serial ordering in both language production and WM research [123].

Memory mechanisms such as chunking play a significant role in language processing. The cognitive system uses chunking to rapidly compress and recode linguistic input, which helps in the retention and processing of information. This mechanism influences linguistic structure and aids in language acquisition and processing [124].

2.4.5.2 Declarative and Procedural Memory

Long-term memory includes both declarative and procedural memory systems. Declarative memory stores factual knowledge and vocabulary, while procedural memory supports the learning and use of grammatical rules [125]. Declarative memory involves associative memory systems in the temporal lobe, while procedural memory involves the frontal/basal-ganglia circuits [126].

The declarative/procedural model distinguishes between the mental lexicon (memorized words) and mental grammar (rules for structuring words). The model ties the lexicon to associative memory systems in the temporal lobe and grammar to procedural memory systems in the frontal/basal-ganglia circuits. This dual-system approach emphasizes the distinct memory mechanisms involved in different aspects of language processing [125].

2.4.5.3 Hippocampal Memory

Hippocampus is another major component of the memory system in the brain associated with episodic and declarative memory. The hippocampal declarative memory system is essential for relational binding and representational flexibility, processes that are crucial for the real-time integration and use of language. This system supports the generation of novel and complex utterances and the incremental integration of multiple sources of information during language comprehension [127].

Sequence Processing

The hippocampus is critically involved in the encoding, retrieval, and disambiguation of sequences of events. This role is fundamental to episodic memory, where the temporal order of events must be preserved to construct coherent memories. Research shows that hippocampal lesions impair the ability to remember the order of events while sparing the recognition of individual events [128]. This suggests that the hippocampus supports the binding of events into sequences, which is essential for understanding complex patterns and making predictions based on past experiences [129].

Hippocampal neurons, such as place cells and time cells, provide a neural basis for sequence processing. Place cells, which fire when an animal is in a specific location, and time cells, which fire at specific moments during a temporal sequence, together support the spatiotemporal context of experiences [130]. During navigation, the hippocampus generates sequences of place cell activations that represent paths through the environment, facilitating spatial memory and planning [129].

Language Processing

The hippocampus also plays a significant role in language processing, particularly in the integration and retrieval of linguistic information. The hippocampus contributes to the flexible use of language by supporting relational memory processes, which are crucial for understanding and generating complex sentences [127]. This is evident in its involvement in tasks that require the association of words and meanings within specific contexts, aiding in the construction of coherent narratives and the understanding of context-dependent meanings [131].

Functional MRI studies have shown that hippocampal activity increases during tasks involving the learning and retrieval of new word sequences and grammatical structures. This activity reflects the hippocampus's role in the formation of new linguistic memories and the retrieval of previously learned sequences, highlighting its importance in language acquisition and processing [132]. Furthermore, the hippocampus interacts with other brain regions, such as the prefrontal cortex and Broca's area, to support the hierarchical and sequential aspects of language. These interactions facilitate the encoding of syntactic structures and the maintenance of linguistic information over time, enabling complex language comprehension and production [110].

The hippocampus plays a significant role in establishing semantic associations between unrelated words. This associative memory function is critical for linking words and their meanings, facilitating complex language processing tasks [133] . The hippocampus is crucial for encoding and retrieving sequences of events, which is fundamental for episodic memory. This ability supports the temporal and contextual aspects of language, enabling the recall and construction of coherent narratives [128].

The hippocampus is also involved in processing the feedback of one's own speech, particularly in detecting mismatches between expected and actual sensory consequences of speaking. This role supports the prediction and adjustment processes necessary for fluent speech production [134].

Hippocampal theta oscillations, which are crucial for memory function, also track the amount of contextual linguistic information during sentence processing. This suggests that the hippocampus actively contributes to language by relating incoming words to stored semantic knowledge [131]. Friston and Buzsaki [135] argues that the brain, including the hippocampus, forms "good enough" models of its environment to facilitate prediction [135]. This applies to language, where the hippocampus helps predict upcoming words and their order. Friston and Buzsaki also posit that the hippocampus efficiently processes sequences by separating "what" (semantic information) from "when" (temporal order), analogous to the separation of "what" and "where" in spatial navigation [135].

Learning language-like rules involves the interaction between the hippocampus and the prefrontal cortex. During artificial language acquisition tasks, increased proficiency is associated with decreased hippocampal activity and increased activity in the prefrontal cortex, indicating a shift from hippocampal-dependent learning to language-related processing [132].

Counter Evidence Against Involvement in Language and Syntax

However, evidence from case studies presents a more nuanced view of the hippocampus's role in language. The famous case of H.M., who underwent bilateral medial temporal lobe resection including parts of the hippocampus, provides critical insights. Despite his surgery, H.M.'s lexical and grammatical abilities remained largely intact, indicating that these brain regions are not solely responsible for grammatical processing [136]. However, further studies revealed that H.M. had difficulties with comprehending complex sentences, particularly those with multiple clauses. MacKay et al. reported that while H.M. could produce grammatically correct sentences, he often struggled with interpreting sentences that required contextual understanding or involved less frequent meanings, tending to rely on familiar, high-frequency interpretations instead [137].

This evidence suggests that while the hippocampus might not be essential for basic grammatical processing, it may play a significant role in handling more complex and context-dependent linguistic

structures. The involvement of the hippocampus in processing longer and more grammatically intricate sentences aligns with the hypothesis that an external memory system capable of preserving sequential relations is necessary for managing longer and more complex language tasks. In contrast, recurrent memory systems that do not necessarily track longer temporal context might suffice for shorter and simpler sequences, but might fail at longer and more syntactically complex examples akin to the test cases used in our simulation experiments.

These findings highlight the potential role of the hippocampus in supporting language processing tasks that require the integration of extensive sequential and contextual information, bridging the gap between simpler grammatical processing and the complex hierarchical structures characteristic of human language.

Overall research shows that memory plays a complicated but important role for language capabilities. Various and complex mechanisms are involved during language processing. One important factor to point out for the scope of this thesis is the working memory involvement to be able to temporarily store representations for syntactic and semantic processing, another important mechanism is the involvement of hippocampus which has the capacity to provide spatiotemportal relations between memory items which might be crucial for tracking some dependencies in time.

2.4.6 Memory, Unification, Control (MUC) Model of Neurobiology of Language

The Memory, Unification, Control (MUC) model [138, 3] is a neurobiological model of language that aims to address shortcomings of earlier models by considering the interactions of various brain regions. The model proposes that three functional components contribute to language processing.

- **Memory:** This component represents the linguistic knowledge, such as phonology, morphology, and syntax, that is encoded in neocortical memory structures, primarily in the temporal cortex and the angular gyrus in parietal cortex. This component is considered language-specific.
- **Unification:** Unification is the process of combining these linguistic building blocks into more complex structures to create meaning. The model identifies syntactic, semantic, and phonological unification processes. This process is thought to occur primarily in Broca's area and adjacent cortex in the frontal lobe. Notably, the specific area within the left inferior frontal cortex that is activated appears to depend on the type of information being unified.
- **Control:** This component encompasses executive control functions related to language use, including selecting the appropriate language for a given context, managing turn-taking during conversation, and directing attention to the most important information in the input. Control processes involve the dorsolateral prefrontal cortex, the anterior cingulate cortex, and areas of the parietal cortex involved in attention.

The MUC model emphasizes that language functions are not confined to specific brain regions, but are instead carried out by dynamic networks of brain areas. Each node within these networks interacts dynamically with other nodes to perform specific functions. The model highlights the extensive connectivity of the language network, which includes regions in both the left and right hemispheres, with a greater degree of lateralization observed for language production. In addition to the arcuate fascicu-

lus, other fiber bundles, such as the superior longitudinal fasciculus, extreme capsule fasciculus, and uncinate fasciculus, connect frontal regions with temporal and parietal regions.

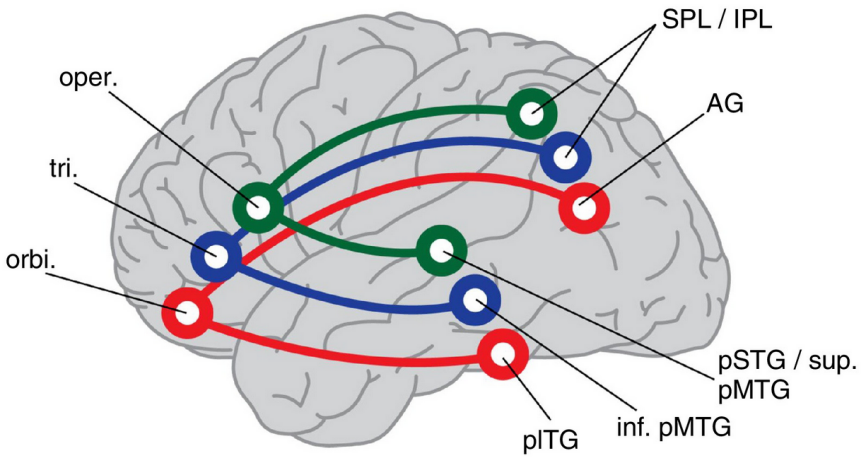


Figure 11: Topographical functional connectivity of the left inferior frontal, inferior parietal, and temporal regions, reflecting the tripartite nature of language processing (phonology, syntax and semantics) (Hagoort, 2014, p. 139) [3]

The MUC model offers a more nuanced perspective on the roles of Broca's area and Wernicke's area than earlier models. While these regions are still considered central to language processing, the MUC model emphasizes a different division of labor and the importance of their interactions with other brain areas. The model posits that Broca's area plays a central role in both syntactic and semantic unification, challenging previous proposals that limit its function to specific syntactic operations or hierarchical processing. It suggests that Broca's area contributes to compositional and decompositional operations across different levels of language processing, including both comprehension and production.

Furthermore, the MUC model posits that language processing goes beyond simply decoding the meaning of individual words and sentences. For instance, it suggests that the inference of speaker meaning involves the Theory of Mind (ToM) network. This network, which includes the temporoparietal junction (TPJ) and medial prefrontal cortex (mPFC), is thought to be crucial for understanding the intentions and beliefs underlying language use, such as indirect requests.

To sum up, the MUC model offers a more dynamic and interactive view of language processing compared to traditional models by emphasizing the interconnected nature of the brain regions involved. It suggests that different brain areas, each specialized for particular functions, work together in a complex interplay to support the full range of human language abilities..

2.4.7 Neural Network Mechanisms

2.4.7.1 Recurrent Neural Networks

Recurrent Neural Networks can store previous activity or previous context at each level.

In theory, RNNs are able to keep track of arbitrarily long-term dependencies, but in practice the linear distance of the items in the sequence has a detrimental effect on the performance mainly because of the vanishing gradient problem. Also recurrent neural networks pass the same memory along layers squishing multiple dependencies in one representation which makes it hard to track. Mechanisms like forget gates in LSTMs help but still don't solve longer dependencies.

2.4.7.2 Transformers

Transformers [54] are effective deep learning mechanisms that have successful applications for many NLP tasks. They usually use encoder decoder structures creating a smaller but meaningful representational space and generate sequences back from those representations.

Currently state of the art dependency parsing applications are based on transformer models. [**?**]. It was shown that some attention heads are sensitive to dependencies during this process.

Transformer models attack the dependency problem by processing all inputs at once. Since they use self attention they do not need to represent the current status of the sequence since they have access to all sequences at any point. This way they do not directly process sequences. So there is no real distance between any of the words since attention is calculated for all combinations. This makes the operation computationally expensive and makes the input size hard to scale. Still, the dependencies it can detect are limited by the input size of the model. And it is harder to relate them to neuroscientific mechanisms because of the sequential nature and information access limitations of the human brain at a given time. It is also shown that self-attention is not suitable for modelling periodic regular languages or basic recursion [83].

2.4.7.3 What is missing in neural network models?

In terms of information processing, humans currently have some important differences. Humans can process longer dependencies. There is no limit to how long the dependencies and sequence can be (competence) other than memory constraints (performance). They can successfully merge and embed things in other things and represent them. They also require much less input for learning. Once some properties of an item are known it can be used successfully. For example if some word is known to be a noun all previous rules of syntax can easily be applied to it without any exposure to the world in the real sense. Lastly, humans do not need to remember the exact utterance of words. They can understand long sentences without remembering the utterance of the whole sentence. Sentences are represented in a different and compressed way so that the reconstruction can be noisy or can be a different sequence with a similar meaning.

2.5 Model Overviews

2.5.1 Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks (RNNs), address key challenges in learning long-term dependencies within sequential data. LSTMs are specif-

ically designed to mitigate the vanishing gradient issue common in traditional RNNs, thus enhancing their capacity to process data with extended sequential dependencies.

The distinctive feature of LSTM networks lies in their memory cell composition, which includes three integral gates: input, forget, and output. These gates collectively manage the information flow within the cell, thereby regulating the cell state across time intervals. The input gate controls the integration of new information into the cell state, the forget gate determines the retention or removal of information, and the output gate influences the portion of the cell state to be outputted at each time step.

This architectural framework empowers LSTMs to retain information over prolonged periods, proving crucial for applications requiring comprehension of extended sequence contexts, such as in natural language processing, speech recognition, and time-series analysis. The efficacy of LSTMs in these domains highlights their capability to model complex temporal patterns and dependencies. [139]

Within the scope of the thesis, LSTMs provide a pertinent model for examining the encoding, processing, and retention of temporal patterns in neural networks. Their applicability in modeling long-range dependencies in sequential data could offer insights into analogous memory processes in cognitive systems. [139]

LSTMs approach the sequence learning problem by updating its internal memory mechanisms using specific gates. Not being a memory network, it lacks the ability to directly store specific representations and recall them. The difference in how memory is used will be used as a comparison point for how well artificial grammars are learned and generalized.

LSTMs were previously shown to be able to perform well in regular and context-free grammars [140]. This will provide a good baseline for a capable neural network that does not have the capacity to directly store items to an external memory and recall them.

2.5.2 Legendre Memory Units (LMUs)

Legendre Memory Units (LMUs) constitute an approach within recurrent neural network (RNN) architectures, aimed at addressing specific challenges related to memory capacity and temporal data processing. Unlike standard RNNs, which are typically optimized for handling short to medium-range dependencies, LMUs are engineered to process more extended sequences and complex patterns.

The LMU employs a memory cell that maintains information over long periods using a set of resources that is relatively compact. This memory cell functions via a system of coupled ordinary differential equations (ODEs). These ODEs are tasked with orthogonalizing the continuous-time history of input signals and utilize the Legendre polynomial basis for mapping these inputs onto temporal windows.

In the architectural layout of the LMU, each layer is composed of individual hidden states and memory vectors. This design allows for the nonlinear transformation of temporal data over time while simultaneously updating the memory content. A notable aspect of this architecture is the separation of the hidden state's dimensionality from the memory size, which contributes to the model's scalability and adaptability.

When compared to other RNN models, such as Long Short-Term Memory (LSTM) networks, LMUs display certain advantages, particularly in chaotic time-series predictions and long-term memory man-

agement. These include potential enhancements in memory capacity and reductions in both training and inference durations. The LMUs' ability to manage long temporal dependencies effectively is attributed to their unique structural composition, facilitating adaptive time-step modulation and the acquisition of scale-invariant features.

However, LMUs do have limitations, especially regarding their approach to data storage and retrieval in the context of long sequences. Unlike specific memory network architectures, LMUs do not explicitly store and retrieve data in a manner akin to conventional memory systems. This limitation can be consequential in scenarios that demand precise historical data recall over very long sequences.

In terms of biological plausibility, LMUs offer an interesting perspective that parallels certain cognitive processing mechanisms in biological neural networks. It is crucial to recognize, however, that LMUs, while potentially reflective of some biological processes, are a computational model and do not fully encapsulate the complexities inherent in biological memory systems.

LMUs performance in sequence processing tasks and ability to handle long sequences combined with their biological plausibility presents it as a notable candidate to see the performance in artificial grammar tasks. [141]

2.5.3 Neural Turing Machines (NTMs)

Neural Turing Machines (NTMs) proposed by Graves et al. [38], represent a significant advancement in the field of neural networks, integrating principles of Turing Machines with modern neural network architectures. This integration yields a model that encapsulates both a neural network controller and a memory matrix, allowing it to perform complex data processing and storage tasks. The NTM's architecture is unique in its incorporation of a memory bank, which interacts with the neural network controller through selective read and write operations.[38]

Components of the NTM are:

- Neural Network Controller: The central component that processes input and output vectors and interacts with the memory matrix. It plays a key role in determining the behavior of the NTM, including how it reads from and writes to the memory.
- Memory Matrix: A structured memory storage that the controller accesses. It consists of a number of memory locations, each capable of storing a vector of data.
- Read and Write Heads: Mechanisms through which the controller interacts with the memory matrix. The read head retrieves information from memory, while the write head updates or adds new information.
- Content-Based Addressing Mechanism: This allows the NTM to focus on memory locations based on the similarity of their content with a specific query, enabling efficient data retrieval.
- Location-Based Addressing Mechanism: This addresses memory locations based on their location rather than content, useful for tasks where the relationship or sequence of information is important.

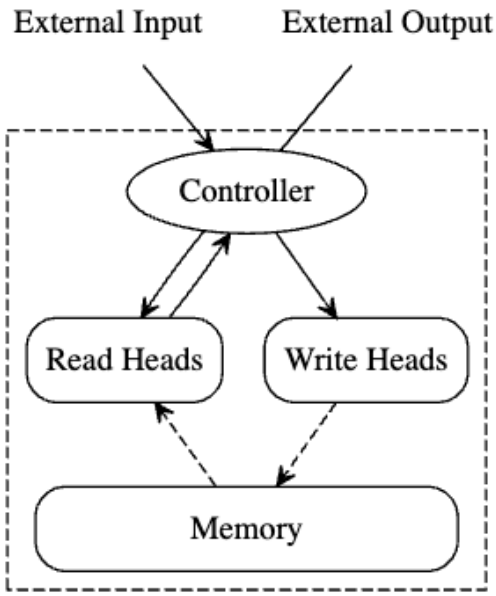


Figure 12: A figure showing the general architecture of NTMs

- Attentional Focus Mechanism: Governs the read and write operations by determining the degree of interaction with various elements in the memory matrix, allowing the NTM to operate on a specific part of the memory.
- Erase and Add Operations: Used during the write process, where the erase operation resets certain memory elements, followed by the add operation that updates these elements with new data.

[38]

The memory matrix of an NTM is structured into N memory locations, each containing an M vector. The read and write operations involve a vector of weightings over these locations, allowing the NTM to focus attentively on specific memory segments. This attention mechanism is crucial for the NTM's functionality, enabling it to engage with different memory areas to varying degrees, thereby optimizing data storage and retrieval processes.

One of the key features of NTMs is the dual addressing mechanisms they employ: content-based addressing and location-based addressing. Content-based addressing allows the NTM to focus on memory locations based on the similarity between their contents and a given query. This is particularly useful in retrieval tasks where the controller needs to approximate a part of the stored data. Location-based addressing, on the other hand, is crucial for problems where the content of a variable is arbitrary, but the variable requires a distinct address or identifier. This form of addressing is vital for tasks involving variable manipulation, as seen in arithmetic operations.

The NTM architecture offers flexibility in terms of its parameters, such as the size of the memory, the number of read and write heads, and the range of allowed location shifts. The choice of neural network used as the controller, either recurrent or feed-forward, is a significant architectural decision.

A recurrent controller, like LSTM, has its internal memory, which can complement the larger memory matrix. In contrast, a feed-forward controller might provide greater transparency in the network's operations, as it allows for more straightforward interpretation of the patterns of reading from and writing to the memory matrix.

NTMs are an example of a more general category of memory networks. Memory Networks, a category of neural network architectures, are designed to incorporate explicit memory components, enabling them to store, access, and utilize past experiences or information. This integration of memory into neural networks marks a significant evolution from traditional models, enhancing their capabilities in handling a wide range of tasks, especially those requiring retention and manipulation of information over time. This capability of explicitly storing and recalling of items can be equivalent to the stack memory mechanism of the push-down automata or the tape in the Turing machines.

In the original Neural Turing Machine (NTM) paper [38] Graves et. al, demonstrated that NTMs perform exceptionally well across a variety of tasks, outshining other recurrent neural networks (RNNs). Key tasks showcasing NTM's superior capabilities include the Copy Task, Repeat Copy Task, Associative Recall, and Priority Sort Task. These tasks highlight NTMs' advanced proficiency in long-term memory retention, nested function execution, complex data structure manipulation, and algorithmic sorting. This enhanced performance is largely attributed to NTMs' unique architecture, which combines neural network controllers with an external memory matrix, and their ability to effectively handle both content-based and location-based addressing. This makes NTMs more adept at tasks requiring dynamic memory usage and intricate data processing, a significant step up from the capabilities of standard RNNs.

This makes the NTMs a good candidate to examine how neural networks can process and learn sequences using external memory mechanisms for artificial grammars of various complexities

2.5.3.1 Attention Mechanism of NTM

The attention mechanism in Neural Turing Machines (NTMs) dictates how the network interacts with its memory, allowing it to read from or write to specific memory locations selectively. This process, referred to as an "introspective attention model," guides the network's focus within its memory. The mechanism functions by enabling the controller to generate vectors that determine a weighted distribution across rows, representing memory locations, within the memory matrix. [38]

Memory networks utilize two primary addressing mechanisms: content-based and location-based addressing. Content-based addressing directs the network's focus to memory locations with content similar to a key vector produced by the controller, akin to how search engines retrieve information based on keywords, much like Hopfield networks. Location-based addressing, on the other hand, accesses memory locations based on their position within the memory matrix, which is crucial when the specific content is irrelevant but its location is vital, such as in arithmetic operations. This mechanism employs a shift operation to move a weighting across the memory, allowing for sequential access or random jumps between locations, visualized as moving a spotlight across a stage.[38]

Combining Addressing Mechanisms By combining these addressing methods, NTMs achieve versatility in how they interact with memory[38]:

- Using only content keys treats the memory like an associative map, directly linking keys and content.
- Combining content keys and location shifts enables treating sections of memory as arrays, where the key pinpoints the beginning and shifts index into the array.
- Relying solely on location and ignoring content allows for iteration through addresses, akin to a list iterator.

This multifaceted interaction with its memory empowers the NTM to tackle intricate tasks requiring the manipulation and storage of data, distinguishing it from other attention-based memory systems.

Potential Parallels with Human Brain Function The attention mechanisms in NTMs can be considered analogous to certain cognitive processes observed in the human brain:

- **Content-Based Addressing and Semantic Memory:** The content-based addressing in NTMs may share some functional similarities with how the human brain retrieves semantic memory. Semantic memory involves general knowledge about the world, which is not tied to specific experiences but rather involves retrieving contextually relevant information [142]. In the brain, this process appears to involve areas such as the hippocampus and anterior temporal lobes, which may facilitate the integration of contextual information [143]. This could be seen as somewhat similar to how NTMs use content keys to retrieve information based on similarity, regardless of its location in memory.
- **Location-Based Addressing and Spatial Memory:** The location-based addressing in NTMs might be likened to certain aspects of the brain's spatial memory processes. In the brain, spatial memory, which involves remembering the locations of objects and navigating environments, is associated with hippocampal function and place cells [144]. This could be seen as loosely analogous to how NTMs can access memory locations based on their position within the memory matrix.
- **Temporal-Spatial Processing:** The brain appears to handle temporal sequences in a manner that shares some characteristics with its processing of spatial sequences, a feature that might be somewhat reflected in the NTM's ability to process sequential information. The hippocampus, important for both spatial navigation and the sequencing of events over time [145], demonstrates a versatility that could be seen as having some parallels with the NTM's combined use of content-based and location-based addressing.
- **Path Integrity in Memory:** The NTM's ability to combine content keys and location shifts to treat sections of memory as arrays could be seen as having some conceptual similarities to how the brain uses path integrity in spatial learning. In the brain, sequentially encountered objects appear to form a coherent path that aids memory, suggesting that temporal sequencing may enhance spatial memory [146]. This might be viewed as somewhat similar to how NTMs can use a content key to identify the beginning of a sequence and then use shifts to index through it.

These potential parallels suggest that the attention mechanisms in NTMs might provide an interesting computational analog to certain cognitive processes in the human brain, particularly in how information is stored, retrieved, and manipulated in memory. However, it's crucial to emphasize that while

these similarities are intriguing, the human brain's complexity far exceeds current artificial neural network models. These parallels should be interpreted cautiously and viewed as points of conceptual similarity rather than direct functional equivalences.

In summary, the literature illustrates significant advancements in both cognitive computational models and deep learning architectures, particularly in their capacity to simulate aspects of human cognition. However, a persistent gap remains in bridging high-level cognitive processes with neuroscientific findings, especially in the context of memory networks and their biological plausibility. While traditional connectionist models have provided valuable insights into lower-level cognitive functions, they often fall short in modeling complex, hierarchical cognitive tasks that require a nuanced understanding of memory and sequence processing. This thesis aims to address this gap by investigating neurocomputational models, particularly focusing on the application of memory networks like Neural Turing Machines (NTMs) and Legendre Memory Units (LMUs), and their potential to model cognitive syntactic and rule learning functions in a manner that is both computationally effective and connectionist way. By integrating insights from

language research and neuroscience with neural network architectures that aims to have computational parallels to human cognitive processes, this work aspires to contribute to the development of more comprehensive models that .

In summary, the literature reviewed in this chapter highlights the considerable advancements in cognitive computational models and deep learning architectures, yet it identifies a gap in effectively integrating memory mechanisms within connectionist networks to model the some aspects of human cognition such as rule learning and syntax. While traditional connectionist models have been successful in simulating lower-level cognitive functions, they often struggle with higher-level processes, particularly those involving complex, hierarchical structures like those found in human language.

Humans possess a unique ability to process hierarchical structures in language, a capacity that is deeply linked to advanced memory systems in the brain, such as those involving the hippocampus and Broca's area. These neural mechanisms enable the real-time integration of sequential and hierarchical information, which is essential for the recursive and context-dependent nature of human syntax—a feature that current computational models struggle to replicate.

Despite the introduction of attention mechanisms in models like Transformers, which have improved the ability of deep learning architectures to manage sequence processing, these models still fall short in mimicking the nuanced memory processes observed in the human brain. Augmented memory networks such as Neural Turing Machines (NTMs) present promising developments in this area, offering potential pathways to integrate memory mechanisms that can handle more rule based tasks. However, their alignment with the biological plausibility of human memory systems remains an ongoing challenge.

This thesis seeks to address this critical gap by investigating how neural network models with certain memory structures can be refined to more accurately simulate the memory mechanisms that underpin human syntactic functions. By focusing on the integration of memory mechanisms within connectionist networks that shows similarities to some aspects of human memory, this research aims to develop models that better capture the intricacies of human syntax and memory, advancing our understanding of cognitive processes and bringing computational models closer to reflecting the complexities of the human brain.

CHAPTER 3

METHODOLOGY

3.1 Introduction to Methodology

The methodology adopted in this research involves training various machine learning models on procedurally generated artificial grammars. The focus is on formalization and syntactic content while providing examples that can be easily understandable and interpretable.Similar to comparative studies on the grammatical capabilities of different biological species, artificial grammar learning tasks were employed to compare the performance of various neural network models in learning artificial grammars of differing syntactic complexity.

This section explains the systematic processes and experimental designs employed to investigate the learning and processing of artificial grammars by various algorithms. These experiments are simulations of the artificial grammar learning processes using the framework. The overarching objective of these methodologies is to explore the computational requirements of various neural networks to learn and predict certain grammatical structures. In doing so, the study aims to shed light on the intricate dynamics between artificial intelligence and human-like language processing abilities.

Machine learning models, such as Neural Turing Machines (NTMs), Long Short-Term Memory (LSTM) networks, and Legendre Memory Units (LMUs), have been selected for their unique architectural traits, their differences in handling memory and sequences and their potential to mimic cognitive functions related to learning artificial grammars and rules of artificial grammars. The procedural generation of artificial grammars aims to provide a diverse range of syntactic structures, challenging these models with the complexity and variability akin to natural languages.

Each model's ability to comprehend, learn, and generalize the grammatical rules will be assessed through evaluation metrics, providing insights on how well they can perform on certain sequence lengths of different grammars and how well they can generalize beyond their sequence lengths provided in training sets. Moreover, the study will attempt to uncover the degree to which these models can predict unseen data and the extent of their memory recall capabilities, which are essential for learning the rules of certain grammars.

The interpretability of model predictions is important; therefore, considerable attention is devoted to the explainability of each model's learning process. This is accomplished through a combination of visualization techniques and an in-depth analysis of model parameters, which facilitate an understanding of how artificial grammars are internally represented by different neural network architectures.

A methodical approach ensures that the experiments/simulations conducted are replicable and the results are statistically valid. This involves a clear definition of the grammars, a consistent training methodology, and an interpretable evaluation process.

In summary, this methodology section serves as a foundation for the modelling experiments that follow. It explains the rationale behind the choice of machine learning models, the design of the artificial grammars, and the evaluation framework. The ultimate aim is to contribute meaningful insights to the field of computational cognitive models and provide a framework for understanding the computational and structural requirements that lie beyond learning and generalizing grammars.

3.1.1 Simulation Experiment Setup Overview

To this end, the experimental setup is structured around a series of controlled scenarios in which simulations are performed with various neural network architectures that are trained and tested with well-defined artificial grammars. This design permits a granular analysis of the connectionist computational mechanisms that process grammars in a simplified linguistic environment. Such an approach is particularly beneficial for isolating specific variables of interest, free from the confounding factors present in natural language processing like semantics and context.

The selection of artificial grammars, namely state machine (Reber grammar), A_nB_n Grammar, and Mirror Grammar, is premised on their diverse complexity and the distinct cognitive strategies they necessitate. The State Machine (Reber grammar) provides a basis for understanding pattern recognition, the A_nB_n Grammar introduces the challenge of nested dependencies, and the Mirror Grammar examines symmetry detection and working memory utilization. A mildly context sensitive grammar (WW^R_WW^R grammar) is also introduced as a more complex grammar that approximates the computational complexity of human languages.

A probabilistic state machine with stack support serves as the implementation framework. It not only emulates the unpredictability of natural language but also facilitates the examination of error-driven learning processes. The input-output format is specifically designed to capture all possible outcomes, thereby embedding a predictive encoding mechanism within the language model. This format ensures that each output is not only interpretable but also conducive to subsequent computational analysis.

Memory considerations are of paramount importance in the encoding process. The methodology accounts for both the capacity of working memory and the role of long-term memory in sequence learning. These considerations influence the complexity of the grammars and the structure of the input-output sequences.

The rigour of this methodological approach is intended to contribute to the existing body of knowledge on artificial grammar learning. It offers a comprehensive framework for examining the fundamental components of language acquisition and the cognitive processes that underpin it. The ensuing sections will provide an in-depth exposition of each aspect of the methodology, paving the way for a detailed discussion of the experimental results and their implications.

3.2 Models Overview

To dissect the nuances of artificial grammar learning, this research employs a range of machine learning models, each with distinct architectural features and learning capabilities. The models are selected based on their proven efficacy in tasks that require pattern recognition, sequential data processing, and memory utilization—key components in language comprehension and production.

The models used in the simulation experiments and their mechanisms are explained in Section 2.5. The LSTM model explained in Section 2.5.1 is selected as a recurrent neural network with gating mechanisms to handle longer dependencies. The LMU model explained in Section 2.5.2 is selected as a unique approach to long-term dependency management and their claimed biological plausibility [141]. And lastly NTM model explained in Section 2.5.3 is selected as a flexible neural network with adjustable external memory. The memory mechanism has content based and address based accessing modes. This mechanisms share similarities with human brain function as detailed in Section 2.5.3.1

3.3 Experiment Data

In the "Experiment Data" section, we engage in an in-depth analysis of how the experimental data is formalized and created to train various machine learning models on different classes of grammars, specifically focusing on regular and supra-regular grammars. This includes an exploration of state-machine grammars, such as the Reber grammar, alongside AnBn Grammar and Mirror Grammar. Each of these grammars presents unique structural characteristics essential to our investigation.

A key aspect of the data creation approach is the procedural generation of sequences. This methodology allows precise control over various attributes of the data, including length and complexity. The advantage of this approach is the generation of a virtually unlimited dataset, tailored specifically to our research requirements.

Our focus is primarily on the syntax aspect of these grammars, intentionally excluding semantic factors. This approach is designed to isolate and scrutinize the structural components of language processing in a more controlled and clear-cut manner.

In terms of learning the states within these grammars, this approach deviates from conventional probabilistic sequence generation. Instead, we aim to represent all possible outputs, a strategy that is anticipated to provide deeper insights into the mechanisms of grammar learning and state representation. This way we can ensure that the models learn the actual rules and states of the grammar rather than exploiting 'a way' to generate sequences in that particular grammar.

The training of various models is integral to our study, with each model offering different perspectives and capabilities in processing and learning artificial grammars. Complementing this, we developed tools to automate the training process and facilitate a more nuanced understanding of the results. These tools are expected to not only streamline the research process but also provide a clearer interpretation of the complex interactions between different cognitive functions and their computational models. These tools will be explained in later sections.

3.3.1 Sequence Learning

The experiments are designed as a sequence learning problem that focuses on predicting all possible outputs from sequential inputs. This necessitates a model's capability to both recognize the current state and internalize the governing grammar rules.

A notable limitation in conventional Recurrent Neural Networks (RNNs) is their struggle with long sequences, often attributed to the vanishing gradient problem [147]. To address this, our approach incorporates advanced architectures that have different mechanisms to handle long sequences like Long Short-Term Memory (LSTM) networks, Legendre Memory Units(LMUs) and Neural Turing Machines (NTMs). These architectures are specifically chosen to enhance sequence learning capabilities, overcoming the limitations of traditional RNNs in learning the sequences.

Our methodology is designed not just for computational efficiency but also for biological plausibility, aligning the learning process of neural networks with cognitive function models. This approach serves a dual purpose: improving the neural networks' handling of complex sequences and offering insights into their potential as models of cognitive processes.

The subsequent sections will explore artificial grammar learning in detail, illustrating how these neural architectures can be applied to govern the grammar and rule learning process.

3.3.2 Artificial Grammars

Artificial grammars, central to our investigation, are structured, rule-based systems designed to mimic certain aspects of natural languages. These grammars, though simplified, are instrumental in studying the underlying mechanisms of language processing and acquisition.

The rationale for employing artificial grammars in our experiments, as opposed to natural language, is fourfold. First, they alleviate data limitations inherent in natural language studies. Since sequences in artificial grammars can be procedurally-generated, we have access to an extensive and adaptable dataset. This flexibility is crucial for conducting comprehensive and varied analyses.

Secondly, artificial grammars allow for better control over complexity. By focusing exclusively on grammar and symbolic processing, we can isolate specific cognitive functions without the confounding variables present in natural language. This isolation is key to understanding the discrete components of language processing.

Thirdly, artificial grammars enhance explainability. The computational limitations and capabilities of models become clearer when dealing with a simplified, rule-based system. This clarity is essential for accurately interpreting results and drawing meaningful conclusions about language processing mechanisms.

Lastly, working with artificial grammars reduces the need for larger, more complex network architectures. Since the focus is on grammar rather than semantic representation, the computational demands are considerably lower. This efficiency allows for more focused and in-depth exploration of grammar processing and symbolic reasoning in neural network models.

In summary, artificial grammars provide a controlled, manageable, and clear platform for investigating the intricacies of language processing, offering valuable insights that may be less discernible in the complexity of natural language.

3.3.3 State Machine (Reber Grammar)

The State Machine, exemplified by the Reber grammar, represents a specific type of artificial grammar used extensively in cognitive science research, particularly in studies focused on sequence learning and memory. Reber grammar is a finite state machine that generates strings according to a set of predefined rules, forming a complex network of possible transitions. [148][149]

In the context of neural network research, Reber grammar serves as an ideal testbed for examining how different models handle the challenges of sequential pattern recognition and rule-based generation. Its structured yet complex nature provides a rigorous framework for testing the capabilities of neural networks, particularly in terms of their ability to learn and predict sequences based on an understanding of underlying grammatical rules.

3.3.3.1 Definition of Reber Grammar

The Reber grammar is a type of embedded finite state machine used in cognitive science and computational linguistics. It generates strings based on a set of predefined rules represented by a state diagram. The grammar is known for its complexity and is used to study pattern recognition and sequence processing.

Given the state diagram, the Reber grammar can be shown in the Figure 13

The Reber grammar is defined by a series of transitions between states, each triggered by specific input characters. The transitions in the grammar based on the provided graph are as follows:

- (Begin) $\rightarrow$ B(#0)
- (#0) $\rightarrow$ T(#1) | P(#2)
- (#1) $\rightarrow$ X(#3) | S(#1)
- (#2) $\rightarrow$ V(#4) | T(#2)
- (#3) $\rightarrow$ S(#5) | X(#2)
- (#4) $\rightarrow$ P(#3) | V(#5)
- (#5) $\rightarrow$ E(End)
- (End) $\rightarrow \epsilon$

Each transition is denoted by the current state, followed by the triggering input character and the resulting state. For instance, the transition (#0) $\rightarrow$ T(#1) indicates that from state #0, upon receiving the input 'T', the system transitions to state #1.

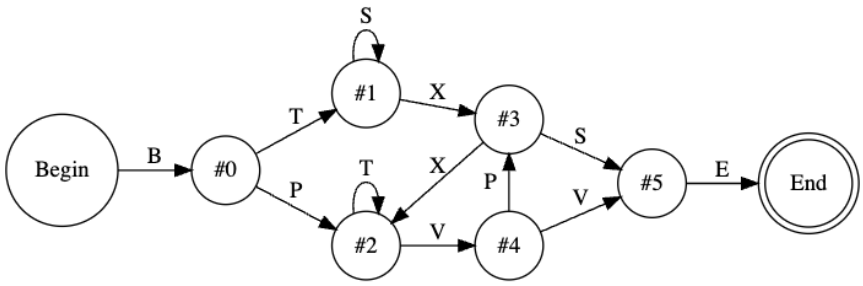


Figure 13: State machine diagram of Reber grammar

Characterized by its distinct structure, the Reber grammar comprises a series of nodes and connecting paths, each path labeled with a specific symbol. The generation of a string involves moving through this network from a starting node to an ending node, with the sequence of symbols along the traversed path forming the output string. This process mimics certain aspects of natural language syntax, making it an invaluable tool for investigating the cognitive processes involved in language acquisition and comprehension.

3.3.3.2 Significance of Reber Grammar in Cognitive Science

The significance of Reber grammar in cognitive science lies in its role as a tool for investigating cognitive processes related to artificial grammar learning (AGL). AGL is a widely used experimental paradigm that explores how syntactic structures are processed and learned [150]. Reber grammar has been widely discussed in the literature and is often used in AGL experiments [150].

Key aspects of Reber grammar and its significance in cognitive science include:

- **Individual differences:** Reber grammar helps researchers understand the cognitive processes that underlie natural language learning by examining individual differences in behavioral patterns despite exposure to the same stimuli [150].
- **Sequence-processing abilities:** AGL experiments using Reber grammar provide a solid basis for investigating sequence-processing abilities in various populations, including infants, monkeys, starlings, and pigeons[150].
- **Transfer in AGL:** Reber grammar has been used to examine the extent to which knowledge of sequential dependencies and patterns of repeating elements is used during transfer in AGL [151].
- **Implicit learning:** Reber's claims concerning the processes that underlie implicit learning of artificial grammars have much in common with widely held views concerning the cognitive unconscious [152].

Overall, Reber grammar is significant in cognitive science because it provides a means to investigate cognitive processes related to artificial grammar learning, individual differences, sequence-processing abilities, and implicit learning. In our simulation experiments the Reber grammar is employed as it

provides a good example of a regular grammar where the internal state representation is not directly related to the input but previous inputs and state transitions should be taken into account to correctly assess the current state thus the possible outputs.

3.3.4 $\mathbf{A}_n\mathbf{B}_n$ Grammar

The $\mathrm{A}_n\mathrm{B}_n$ grammar is a variant of context-free grammar widely utilized in cognitive science and computational neuroscience to explore non-sequential pattern recognition and sequence processing abilities. This grammar generates sequences with an equal number of 'A's and 'B's, starting with A's and followed by an equal number of B's. The complexity lies in the system's capacity to track and match the quantities of 'A's and 'B's in any given order.

3.3.4.1 Definition of $\mathbf{A}_n\mathbf{B}_n$ Grammar

$\mathrm{A}_n\mathrm{B}_n$ grammar is a context-free grammar that generates strings consisting of 'n' occurrences of the character 'A' followed by 'n' occurrences of the character 'B'. For any positive integer 'n', the grammar produces strings of the form $\mathrm{A}_n\mathrm{B}_n$, where A_n denotes 'n' consecutive 'A's and B_n denotes 'n' consecutive 'B's.

Formally, the grammar can be defined as:

$$G = (V, \Sigma, R, S) \tag{1}$$

where,

- V is a set of variables.
- Σ is a finite set of terminals where $\Sigma \cap V = \emptyset$.
- R is a set of production rules.
- S is the start symbol.

The production rules in R are given by:

$$S \to ASB \tag{2}$$
$$A \to a \tag{3}$$
$$B \to b \tag{4}$$
$$S \to \varepsilon \tag{5}$$

3.3.4.2 Significance of $\mathbf{A}_n\mathbf{B}_n$ Grammar in Cognitive Science

Key aspects and significance of $\mathrm{A}_n\mathrm{B}_n$ Grammar in cognitive science can be summarized as follows:

- **Memory and Pattern Recognition:** This grammar challenges the system's memory capacity to track and balance the count of 'A's and 'B's in a non-sequential manner, emphasizing flexible memory management and pattern recognition. Either by counting or storing items in memory.
- **Non-Sequential Grammar Processing:** The A_nB_n grammar necessitates computational mechanisms that can handle unordered inputs, providing insights into the capabilities of neural network architectures in processing non-sequential patterns. [153]
- **Flexible Hierarchical Structure Processing:** This grammar underlines the importance of processing capabilities beyond fixed hierarchical sequences, crucial for understanding the flexibility of human syntactic processing and its emulation in artificial neural networks. [154]
- **Benchmark for Neural Network Models:** Evaluating neural networks like RNNs, LSTMs, and NTMs against the A_nB_n grammar assesses their ability to learn and generalize rules in a non-sequential context, testing the adaptability of these models. [155]
- **Implications for Cognitive Science:** Modeling A_nB_n grammar processing in neural networks can inform cognitive theories, particularly in understanding how cognitive processes manage non-sequential information and flexible syntax structures. [156]
- **Understanding Language Development:** Studies of A_nB_n grammar in computational models can shed light on the developmental aspects of human language, especially on how children comprehend and produce language with flexible and non-sequential syntactic structures. [157]
- **Bridging Neuroscience and AI:** Exploring the processing of A_nB_n grammar in neural network models helps in drawing parallels between artificial systems and biological neural networks, contributing to a more comprehensive understanding of neurocomputational modeling. [158]

3.3.5 Mirror Grammar

The Mirror Grammar is a type of artificial grammar used to study symmetry detection and working memory in cognitive science and computational modeling. It is designed to produce sequences where the second half is a mirror image of the first half, such as ABBA or ABCBA. This structure challenges models to learn and recognize symmetrical patterns, testing their memory and pattern recognition capabilities.

A specific version of the mirror grammar where there is a dedicated middle character 'M' will be used in this study. This will allow the sequence processing mechanism to know where the middle of the sequence is and predict correctly based on the rules. The alternative will not be feasible for learning purposes because all possible characters can be output at any given time. This would render the prediction of next sequences meaningless.

3.3.5.1 Definition of Mirror Grammar with Central Character

The Mirror Grammar with a central character 'm' is a formal grammar that generates palindromic strings with 'm' as the midpoint. In this grammar, each string starts and ends with the same sequence of characters in reverse order around the central character 'm'.

Formally, the grammar can be defined as:

$$G = (V, \Sigma, R, S) \tag{6}$$

where,

- V is a set of variables.
- Σ is a finite set of terminals where $\Sigma \cap V = \emptyset$.
- R is a set of production rules.
- S is the start symbol.

The production rules in R are given by:

$$S \rightarrow aSa \mid bSb \mid cSc \mid dSd \mid m \tag{7}$$

This implies that each string generated by this grammar is symmetric around the central character 'm', such as 'aamaa', 'bmb', 'abmba', and so on. The grammar is particularly useful in computational models for testing symmetrical pattern recognition and memory processing capabilities.

3.3.5.2 Significance of Mirror Grammar in Cognitive Science

Key aspects and significance of Mirror Grammar in cognitive science include:

- **Symmetry Recognition:** Mirror Grammar is instrumental in studying how cognitive systems process symmetrical patterns, a fundamental aspect of human perception and cognition.
- **Working Memory Utilization:** The grammar tests a system's ability to temporarily store and manipulate information, reflecting the capabilities of human working memory.
- **Neural Network Modeling:** For neural network architectures, especially those focusing on sequential data processing, Mirror Grammar provides a unique challenge to assess their learning and generalization abilities in recognizing mirrored sequences.
- **Cognitive and Computational Implications:** Insights gained from modeling Mirror Grammar can inform theories in cognitive science, particularly in understanding how the brain processes patterns and manages symmetrical information.

In the context of this research, the performance of Neural Turing Machines (NTMs), Long Short-Term Memory (LSTM) networks, and Legendre Memory Units (LMUs) will be evaluated against the Mirror Grammar. This evaluation will provide a deeper understanding of each model's capabilities in handling symmetrical structures and their potential applications in cognitive modeling.

3.3.6 Mildly Context Sensitive AWA ($WW^R\_WW^R$) Grammar

3.3.6.1 Definition of Mildly Context Sensitive Grammars

Mildly context-sensitive grammars (MCSGs) are a class of formal grammars that extend the expressive power of context-free grammars while maintaining computational efficiency. They were introduced to capture linguistic phenomena that context-free grammars could not adequately describe, such as certain types of dependencies and cross-serial dependencies found in natural languages. [159]

Key features of mildly context-sensitive grammars include:

1. **Increased Expressive Power:** MCSGs can generate languages that are beyond the capability of context-free grammars (CFGs), including some context-sensitive languages (CSLs), but are still constrained to prevent excessive computational complexity [160].
2. **Polynomial Parsing Time:** Despite their increased expressive power, MCSGs can be parsed in polynomial time, making them feasible for practical applications in computational linguistics [159].
3. **Closure Properties:** MCSGs maintain closure under union, intersection, concatenation, and Kleene star operations, similar to CFGs [159].
4. **Well-defined Hierarchy:** They form a well-defined hierarchy of grammars, sitting between CFGs and general CSLs [159].
5. **Mild Non-linearity:** MCSGs allow for limited cross-serial dependencies, which is crucial for capturing certain natural language phenomena that are difficult or impossible to represent with CFGs [161].
6. **Constant Growth Property:** The lengths of terminal strings increase in a linear and predictable manner during derivations, ensuring computational efficiency and restricting the class of languages generated [162]

Common formalisms that fall under mildly context-sensitive grammars include:

- **Tree-Adjoining Grammars (TAGs):** These grammars generate trees instead of strings, allowing for more complex hierarchical structures [163].
- **Head Grammars (HGs):** A type of grammar that focuses on the head-dependent relationships within sentences [159].
- **Linear Indexed Grammars (LIGs):** A formalism that extends context-free grammars by allowing stacks of bounded depth [159]..
- **Combinatory Categorial Grammars (CCGs):** These grammars are highly lexicalized and use combinatory rules to capture dependencies in natural language [164].

MCSGs have been particularly useful in computational linguistics for modeling the syntactic structure of natural languages, enabling the representation of complex linguistic phenomena while remaining computationally tractable [160].

3.3.6.2 Definition of Mildly Context Sensitive AWA or WW^R_WW^R Grammar

AWA grammar or (WW^R_WW^R Grammar as a more common name where W represents a word) is a combinatory categorial grammar that combines Mirror Grammar (WW^R) with a copy grammar WW. This outputs a mirror grammar string then copies the output. The only difference being that a middle character is introduced to make it possible for the machine learning models to know where the copying starts.

In terms of complexity AWA grammar is equivalent to WW^R_WW^R grammar and they can be used interchangeably in the thesis. But to be able to use the capabilities of the already existing framework it is formalized as follows:

Let $G_{\text{mirror}} = (V_m, \Sigma_m, R_m, S_m)$ be the mirror grammar as defined in Section 3.3.5.1 of this document.

We define our mildly context-sensitive grammar G_{MCSG} in terms of the language it generates:

$$L(G_{\text{MCSG}}) = \{xWx \mid x \in L(G_{\text{mirror}})\}$$

Where:

- $L(G_{\text{mirror}})$ is the set of palindromes over $\{a, b, c, d\}$ with a middle marker m.
- W is a special symbol marking the midpoint of the string.
- x represents any string generated by the mirror grammar.

This concise definition captures the essential structure of the AWA (WW^R_WW^R) grammar:

- It generates strings of the form xWx.
- The substrings on either side of W are identical.
- Each substring x is a valid string in the language of the mirror grammar.

Example A valid string in $L(G_{\text{MCSG}})$ could be:

$$abcdmdcbaWabcdmdcba$$

Here, $x = abcdmdcba$, which is a valid palindrome in $L(G_{\text{mirror}})$.

Properties:

1. This grammar is mildly context-sensitive as it can generate non-context-free languages.
2. It exhibits nested dependencies from the mirror grammar component.

3. It shows the duplication language property (xWx), which is beyond the power of context-free grammars.

4. The language maintains polynomial parsing time, characteristic of MCSGs.

3.3.6.3 Definition of AWA-Star or WW^R_WW^R(_WW^R)* Grammar

The AWA-Star grammar, also known as WW^R_WW^R(_WW^R)* grammar, is an extension of the AWA (WW^R_WW^R) grammar defined in Section 3.3.6.2. This grammar allows for multiple repetitions of the WW^Rpattern after the initial WW^R_WW^R structure. We can define this grammar formally as follows:

Let $G_{\text{mirror}} = (V_m, \Sigma_m, R_m, S_m)$ be the mirror grammar as defined in Section 3.3.5.1. We define our extended mildly context-sensitive grammar G_{MCSG^*} in terms of the language it generates:

$$L(G_{\text{MCSG}^*}) = \{xWx(Wx)^n \mid x \in L(G_{\text{mirror}}), n \geq 0\}$$

Where:

- $L(G_{\text{mirror}})$ is the set of palindromes over $\{a, b, c, d\}$ with a middle marker m.
- W is a special symbol marking the separation between copies.
- x represents any string generated by the mirror grammar.
- n is the number of additional repetitions of Wx.

This definition captures the structure of the AWA-Star/WW^R_WW^R(_WW^R)* grammar:

- It generates strings that start with the AWA pattern xWx.
- This is followed by zero or more repetitions of Wx.
- Each substring x is a valid string in the language of the mirror grammar.

Example A valid string in $L(G_{\text{MCSG}^*})$ could be:

$$abcdmdcbaWabcdmdcbaWabcdmdcbaWabcdmdcba$$

Here, $x = abcdmdcba$, which is a valid palindrome in $L(G_{\text{mirror}})$, and $n = 1$ (one additional repetition).

Properties:

1. This grammar remains mildly context-sensitive, generating a more complex non-context-free language than the basic AWA grammar.

2. It exhibits nested dependencies from the mirror grammar component, repeated multiple times.

3. It shows an extended duplication language property ($xWx(Wx)^n$), which is beyond the power of context-free grammars and more complex than the basic AWA grammar.

4. The language maintains polynomial parsing time, characteristic of MCSGs, though the complexity increases with the number of repetitions.

5. This grammar introduces a form of iteration or recursion not present in the basic AWA grammar, potentially making it more challenging for neural networks to learn and generalize.

The AWA-Star WW^R_WW^R(_WW^R)* Grammar provides a more complex challenge for neural network models, testing their ability to recognize and generate patterns with variable-length repetitions while maintaining the mildly context-sensitive properties.

3.3.6.4 Superstate Architecture for Mildly Context-Sensitive AWA Grammar Generation

The generation of mildly context-sensitive AWA (MCSG-AWA or WW^R_WW^R) grammar is implemented using a novel superstate architecture. This approach allows for flexible combination and nesting of different grammar types within a unified framework. Superstates are an extension of traditional finite state machines, where a state can contain its own internal state machine with a defined grammar. These superstates make internal transitions until their inner state machine concludes, after which they can transition to external states. From an external perspective, a superstate appears to make transitions to itself while outputting the results of its inner state machine.

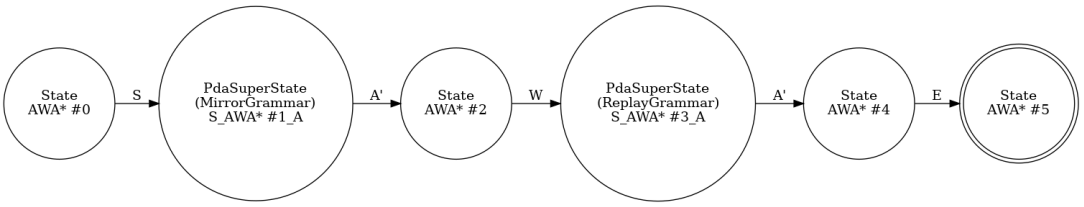


Figure 14: State diagram of AWA grammar with SuperState Architecture

Figure 14 illustrates the superstate architecture for MCSG-AWA grammar generation. The diagram shows how superstates (PdaSuperState) are integrated into the overall state machine structure. The base AWA grammar is implemented as a simple state machine that repeats the character following the 'W' (which is a W token not to be confused with placeholder for word). To incorporate more complex structures, the state responsible for the 'A' transition is replaced by a superstate that produces the output of the MCSG. To enable repetition of the grammar generated by the first superstate, a Repetition Grammar is defined. This grammar exactly replicates the outputs of the first grammar, allowing for iterative structures. In Figure 14, this is represented by the PdaSuperState labeled as (ReplayGrammar).

Similarly Figure 15 demonstrates how the AWA-Star (WW^R_WW^R(_WW^R)*) architecture is generated. It works in a similar way with AWA (WW^R_WW^R) grammar implementation but this time

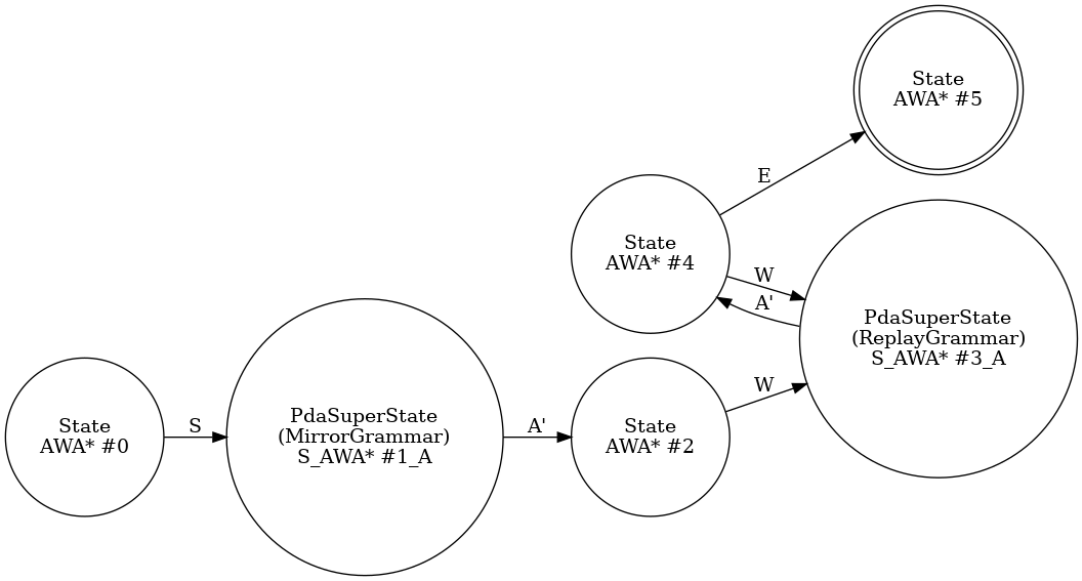


Figure 15: State diagram of AWA-Star grammar with SuperState Architecture

After the State #4 following Replay Grammar Superstate, there is an option to output W and go back to the Replay Super State.

The proposed implementation provides the distinct properties and advantages. They provide modularity where different grammar types can be easily combined and nested. Each of the super states can have their internal components which provide a limitless number of combinations. This provides the flexibility to create more complex grammars from simpler components. It also provides extensibility so that it is easy to add new grammar types as super states without the need to modify existing code or structure.

This implementation approach enables the creation of various Combinatory Categorial Grammars in a flexible and intuitive manner. By leveraging superstates, the system can generate complex mildly context-sensitive structures while maintaining a clear and modular architecture.

3.3.6.5 Pre-training Grammar for Mildly Context-Sensitive AWA (WW^R_WW^R) Grammar

To manage the complexity of the learning process, a pre-training task is implemented before introducing the full AWA grammar (WW^R_WW^R). This pre-training phase is designed to help the model learn the fundamental structures of the inner grammar before tackling the more complex replication patterns. This grammar includes the mirror grammar (WW^R) part without repetition.

As illustrated in Figure 16, the pretraining task consists of a simplified state machine that generates a mirror grammar, followed by a 'W' character, before terminating. This structure allows the model to focus on learning the core mirror grammar mechanics without the additional complexity of the AWA grammar's repetition. The state definitions are the same with Figure 14, but transitions are altered where there is no transition to state #4, instead the state-machine directly ends from there.

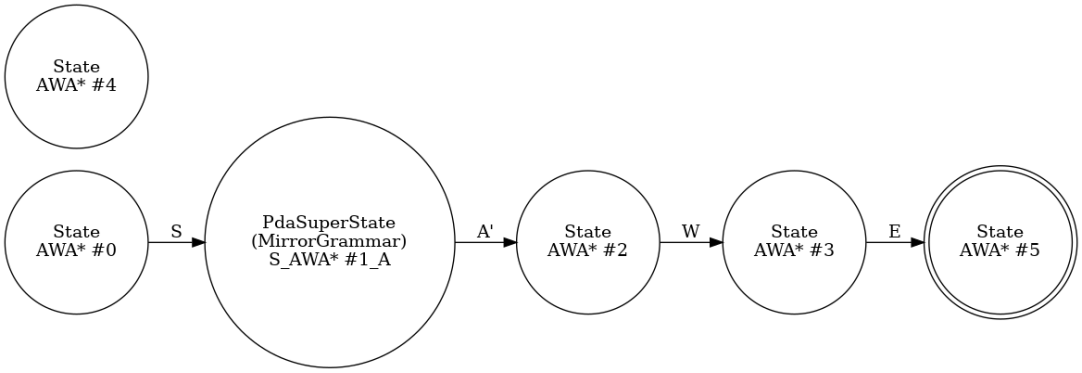


Figure 16: Simplified Mirror Grammar (WW^R) for Pretraining

The pretraining process offers several advantages. It introduces the model to the mirror grammar structure in isolation, allowing for more focused learning increasing the complexity gradually. It allows the model to build a foundation by learning the rules of the inner grammar that will be used later. This is an example of transfer learning where knowledge gained from the pre-training task can be transferred to the more complex AWA grammar.

Models trained on this simplified grammar will subsequently be used as a starting point for fine-tuning on the complete AWA grammar. This approach aims to enhance the model's ability to learn and generalize the more complex structures present in the full MCSG-AWA (WW^R_WW^R) grammar. By implementing this pretraining step, we create a more structured learning path for the model, potentially leading to improved performance and a better understanding of the underlying grammatical structures.

3.3.6.6 Mildly Context-Sensitive Grammar Design

This study uses a mildly context-sensitive grammar (MCSG) that shares features with Combinatory Categorial Grammars (CCGs). We create this MCSG by incorporating a mirror grammar, described in Section 3.3.5, into a larger structure. This structure repeats the mirror grammar's output, separated by a special 'W' character.

Our MCSG, which we call G_{MCSG}, is formally defined in Section 3.3.6.2. It expands on the mirror grammar G_{mirror} by adding new rules. The most important new rule is $S \rightarrow A_{\text{MCSG}} W A_{\text{MCSG}}$, where A_{MCSG} produces strings from the mirror grammar. This creates a language that looks like $\{xWx \mid x \in L(G_{\text{mirror}})\}$, essentially doubling the palindromes generated by the mirror grammar.

We chose this design because it captures key features of mildly context-sensitive languages, especially their ability to handle complex sentence structures and relationships. The mirror grammar part creates nested structures, while the overall design adds a copying feature that context-free grammars can't capture. This combination introduces key grammatical features in our model language, such as multiple agreement and crossed dependencies, which are common in natural languages but can't be represented by simpler grammar systems.

An important feature of our MCSG is that it can be parsed in polynomial time, which means it's computationally efficient. This makes it useful for modeling complex language structures without

becoming too slow to process, striking a balance that's important in computational linguistics and cognitive modeling.

By combining the context-free aspects of the mirror grammar with the agreement patterns from string repetition, our MCSG offers a comprehensive tool for studying complex language structures. It's particularly useful for investigating how neural networks, especially those with external memory, can process and learn the hierarchical and recursive structures found in human language.

The formal definition of G_{MCSG} ($\text{WW}^R\_\text{WW}^R$) in Section 3.3.6.2 captures these properties. This definition not only explains how our MCSG is structured but also helps us understand how it relates to the mirror grammar it's based on and to the broader category of mildly context-sensitive languages.

3.3.6.7 Significance of Mildly Context-Sensitive Grammars in Cognitive Science

Mildly Context-Sensitive Grammars (MCSGs) play a crucial role in cognitive science, particularly in the study of cognitive mechanisms of language and language acquisition. Their significance stems from their ability to model complex linguistic phenomena while maintaining computational tractability.

Key aspects and significance of MCSGs in cognitive science include:

- **Modeling Natural Language Complexity:** MCSGs can represent crossed dependencies and multiple agreements, which are common in many human languages but beyond the capabilities of simpler grammars [165].
- **Alignment with Human Language Processing:** Research suggests that human language processing aligns with the computational properties of MCSGs, providing insights into neural mechanisms of language [166].
- **Competence-Performance Balance:** MCSGs offer a balanced approach to the competence-performance distinction in linguistics, reflecting human ability to process complex language efficiently [165].
- **Language Acquisition Modeling:** MCSGs have been used to model developmental stages of language learning, bridging theoretical linguistics and developmental psychology [166].
- **Working Memory and Language Processing:** The ability to handle crossed dependencies in MCSGs has been linked to working memory capacity in humans, connecting grammatical theory and cognitive psychology [165].
- **Computational Cognitive Modeling:** MCSGs serve as benchmarks for evaluating language processing capabilities of artificial neural networks, assessing how well these models approximate human language abilities [167].
- **Language Universals and Biological Basis:** The formal properties of MCSGs contribute to debates about language universals and the biological basis of language, suggesting potential innate constraints on language learning and processing [165].

Mildly Context-Sensitive Grammars provide a powerful tool for investigating the cognitive processes underlying language comprehension, production, and acquisition, bridging theoretical linguistics, psycholinguistics, and computational cognitive science.

3.3.7 Input-Output Format

The format of input and output is a fundamental aspect of the machine learning training process. This section will explore the details of how inputs and outputs are formalized and why.

3.3.7.1 Encoding

To facilitate the encoding of different grammatical structures, a binary encoding format is employed. In this format, each token from the task-specific vocabulary is assigned a unique position in a vector. The inputs are encoded using a one-hot encoding scheme. This means that for each input token, a vector is created where the position corresponding to that token is set to 1, and all other positions are set to 0. This encoding method is straightforward and effective, especially given that the inputs involve single tokens at each step.

The encoding process differs slightly for each task, depending on its specific vocabulary.

For the State Machine (Reber grammar) task, the vocabulary consists of the tokens B, T, S, X, V, P, E. The binary encoding for some of these tokens is illustrated as follows:

$$B : [1, 0, 0, 0, 0, 0, 0]$$
$$X : [0, 0, 0, 1, 0, 0, 0]$$
$$E : [0, 0, 0, 0, 0, 0, 1]$$

The $\mathrm{A}_n\mathrm{B}_n$ Grammar task utilizes a vocabulary of S, A, B, E. Each sequence in this task begins with the S token and ends with the E token.

Similarly, the Mirror Grammar task includes a vocabulary list of S, A, B, C, D, M, E. In this case, every sequence starts with the S token and ends with the E token, with an M token always present in the middle. Binary encoding for these tasks is performed with a vector whose size is equal to the number of items in the vocabulary.

3.3.8 Experimental Framework Properties

This section explains the critical properties of our experimental framework, detailing the input-output format and the methodology employed in our experiments. The framework has been meticulously designed to ensure that the models learn the underlying algorithms for generating or accepting/rejecting

the grammar, rather than merely capturing statistical correlations between inputs and outputs. Our approach is characterized by a rigorous structure that compels the models to internalize the grammatical rules. This format is crucial in distinguishing between simple pattern recognition and true rule learning. By presenting all possible outcomes at each stage of sequence generation, we force the models to understand and represent the full range of grammatical possibilities, rather than defaulting to the most statistically likely outcomes. In our framework, the rules of the grammar are determined by two key elements:

1. Internal states and inner states that have their own inner grammars (with inner states being required only for mildly context-sensitive grammars)
2. Information shared by the states (In superstate architecture one superstate can copy the output of the sequence generated by other superstate)
3. Possible transitions from these states, which may require elements to be popped from the stack memory in certain grammar types

This structure is crucial because if a state is misrepresented or any rule is missing, one or more of the possible outputs would be incorrect. This property allows us to precisely evaluate whether the models are accurately learning and representing the full grammatical structure. This methodology allows us to evaluate whether the models are genuinely learning the grammar's generative rules or merely memorizing specific patterns. It provides a robust test of the models' ability to abstract and apply grammatical rules to novel situations, which is essential for assessing their potential as neurocomputational models of cognitive functions.

3.3.8.1 Inductive Learning Task

Our experimental framework centers on an inductive learning task. This approach is key to studying artificial grammar learning in neural networks, as it reflects how humans can learn grammars and generalize rules. .Learning a rule involves deriving a general rule from a limited set of examples [168] and using this rule to make predictions about new cases [169].

In our experiments, neural networks receive a finite set of grammatically correct sequences. From these, they must infer the underlying grammatical rules. This process mimics how humans learn language patterns from limited exposure.

The inductive learning approach serves several purposes:

1. It tests if models can generalize beyond memorization
2. It assesses their ability to extract abstract rules
3. It allows comparison of rule-learning across different network types
4. It helps relate model learning to human language acquisition theories

By using inductive learning tasks, we create a robust method to evaluate neural networks as cognitive models. This approach distinguishes between models that merely recognize patterns and those that can understand and apply grammatical rules.

3.3.8.2 Capturing All Possible Outcomes

When training machine learning models for sequence learning, particularly in the context of artificial grammars, it is crucial to ensure that the algorithm learns not just to generate sequences but to understand the rules and produce all possible outcomes at each stage of the sequence. This approach is fundamentally different from training a model to generate random sequences. In the latter case, finding a single pathway to generate sequences might be sufficient for success. However, in our context, representing all possible outcomes at each stage ensures that the model correctly learns the internal states and the rules of grammar.

For instance, in the State Machine (Reber grammar), after the token B, the possible outputs are T and P when the system is in state $\#0$. The output representation in this scenario would include both possible transitions. This representation ensures that the model recognizes the flexibility in the grammar and does not overly specialize in a single pathway. The output encoding in this case would be:

$$\begin{aligned} B, T &: [1, 1, 0, 0, 0, 0, 0] \\ S, X &: [0, 0, 1, 1, 0, 0, 0] \\ E &: [0, 0, 0, 0, 0, 0, 1] \end{aligned}$$

The encoding for B,T represents the possibility that both B or T can be outputs at that time.

Visualisation of inputs and outputs can be seen in Figure 17. The 'Target' diagram represents the expected next tokens given an input tokens shown in 'Inputs' diagram. The 'Outputs' represent what the model predicts as possible next tokens given the input. The input diagram visualizes the sequence of tokens provided to the model from left to right. Each dark box represents a token in the grammar. This allows us to see how the model behaves given each input, what the expected output is and what the actual model output is.

This method of output encoding, where all viable outcomes are presented at each stage, is crucial in ensuring that the learning process is not just about sequence generation but about understanding the underlying grammatical structure. It is through this comprehensive approach that the models are trained to capture the intricacies and variability inherent in each grammar, allowing them to generalize effectively to new, unseen sequences while adhering to the grammatical rules.

This approach also sets the stage for evaluating the models' ability to generalize and handle long-term dependencies, aspects that are critical in the realm of cognitive function modeling and the development of neurocomputational models. The following sections will delve into the specifics of model training, evaluation, and output interpretation, providing a comprehensive view of how these models are developed, tested, and understood in the context of cognitive function simulation.

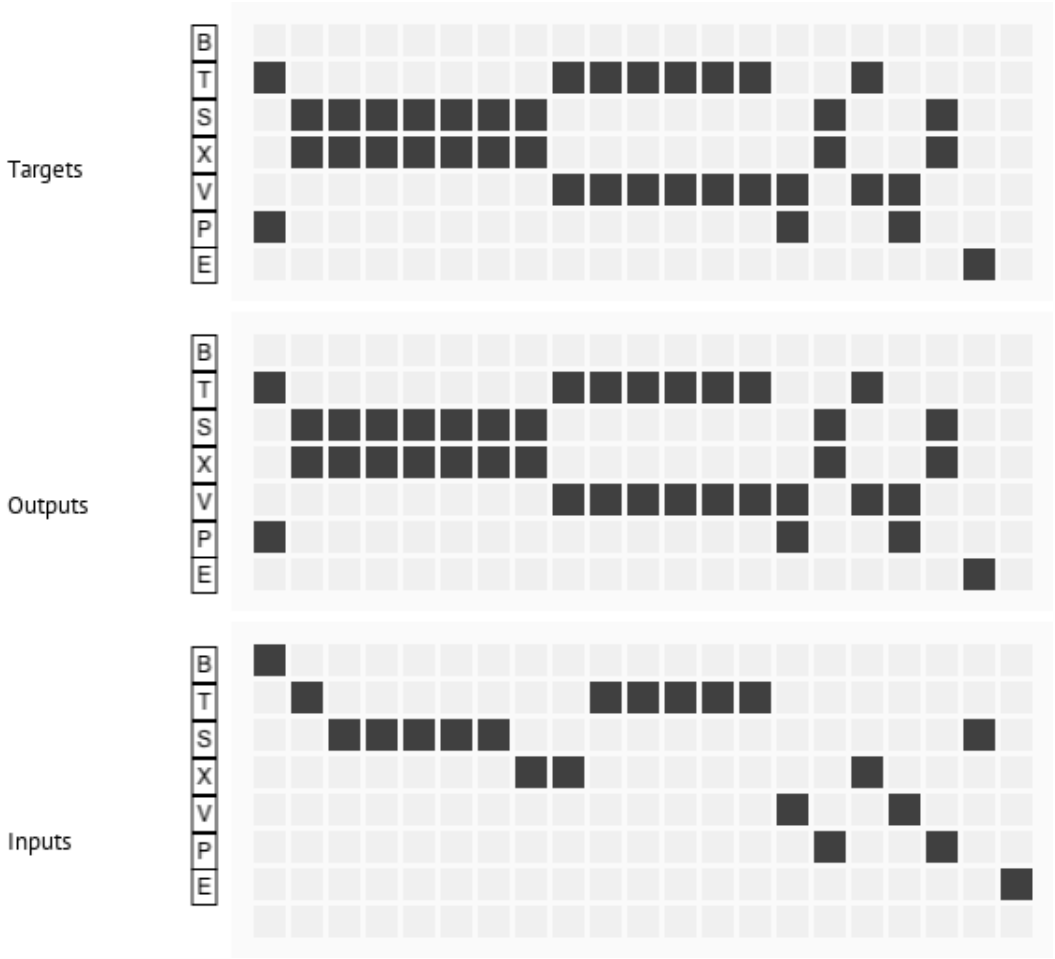


Figure 17: An example sequence for (Reber grammar)

3.3.8.3 Predictive Model Properties

The model is designed to function as a predictive model. All possible next tokens in a sequence are accurately predicted in a generalized model. Through this approach, not just the most common patterns but the full range of grammatical rules is learned. This can serve as a language model for the learned grammars.

3.3.8.4 Supporting Both Grammar Generation And Rejection

Both grammar generation and rejection are supported by our model. Key points include:

- It is not designed as a simple classifier that only accepts or rejects grammars.
- Direct classifier training is avoided, as it's challenging to create a finite set of negative examples that are sufficiently complex [32].

- Training on sequence generation alone is not done, as learning all grammatical rules isn't guaranteed by this method.

For language classification, the following process can be used:

1. The sequence is processed token by token.
2. The sequence is rejected if the next token isn't in the set of possible tokens.
3. If the next token is valid, the process is continued.

In Figure 17 it can be seen that for an input that is in grammar the input of each step was one of the active outputs of the previous step.

For language generation, the following procedure can be used:

1. The initial token is used as a starting point.
2. A random selection is made from possible next tokens.
3. This process is continued until a termination token is selected.

3.3.8.5 Completeness and Soundness

A generalized model is made both **complete** and **sound** by the properties described in Section 3.3.8.4.

- **Completeness:** By using the method in Section 3.3.8.4, all valid sequences in the grammar can be generated by the model. All possible transitions are considered at each step, ensuring no valid sequence is missed.
- **Soundness:** Only grammatically correct sequences are matched by the model output. If the given input was nor present in the possible outputs of the previous input the string can be rejected. This way only valid transitions are selected, eliminating incorrect sequences.

3.3.9 Memory Considerations for Model Training

In this study, we introduce a unified framework to simplify and compare the memory capacities of various neural network architectures, specifically Neural Turing Machines (NTMs), Long Short-Term Memory networks (LSTMs), and Legendre Memory Units (LMUs). The key terms used in this framework are *memory width* and *memory height*, which provide a standardized approach for analyzing and comparing these networks.

- **Memory Height:** Refers to the number of items the network can store. This is directly applicable to NTMs and LMUs but is adapted for LSTMs to represent the complexity of information they can process, rather than a count of distinct items.

- **Memory Width:** Roughly equivalent to the size (dimensionality) of each item stored. This term is directly applicable to NTMs, but for LSTMs and LMUs, it is abstracted for the sake of comparison.

The memory mechanisms of different networks are described as follows:

- **Neural Turing Machines (NTMs):**
  - *Memory Height* (memory_N): Represents the number of memory slots in the NTM.
  - *Memory Width* (memory_M): Corresponds to the vector size of each memory slot.
- **Long Short-Term Memory Networks (LSTMs):**
  - *Memory Height*: The size of the hidden layer, representing the network's capacity to process complex information over time.
  - *Memory Width*: Simplified to a default value of 1, acknowledging the lack of a direct equivalent in LSTMs for memory width.
- **Legendre Memory Units (LMUs):**
  - *Memory Height*: Determined by the number of Legendre polynomials, indicating the variety of temporal patterns that can be remembered.
  - *Memory Width*:The hidden layer size contributes to the network's capacity for processing information, but since many number of Legendre polynomials can be supported, hidden layer size will be used to determine the width.

It is important to note that these terms, particularly *memory width* and *memory height*, are used metaphorically for LSTMs and LMUs to facilitate comparison. They are not direct technical equivalents as in NTMs but are rather conceptual tools designed to standardize discussions across different architectures.

3.4 Experimental Setup

3.4.1 Implementation Details

The deep learning environment was created using the PyTorch library[170]. The LSTM implementation was directly used from PyTorch. NTM [38] implementation was taken from a github repository with a reliable implementation of NTM [171]. The experiment structure is heavily modified to fit the experimental needs in the study. Feed-forward controller implementation is added to the library.

For LMU [141] a PyTorch Implementation is utilized [172].

Experiment wrappers were written for uniformity and being able to use the same experiment structure.

3.4.2 Probabilistic State Machine Implementation

For the data generation infrastructure, we created a state machine implementation that can randomly generate sequences for a defined grammar.

A state machine is defined by states, transition rules and outputs. The outputs are the same with the vocabulary defined for each grammar in Section 3.3.7

3.4.2.1 Generating Sequences

To generate the inputs and the corresponding outputs correctly, we first classify the transitions as short and long transitions. The long transitions are the transitions that do not decrease the distance to the end state. Distance can be defined as the minimum number of transitions to go to the target state. In figure 13 'S' transition at state #1, 'T' transition at state #2 and 'P' transitions at state #4 are long transitions. When generating sequences the probability of long and short transitions are calculated as follows:

$$p_l = \frac{\max 0, N_{target} - N_{generated}}{N_{target}} \quad p_s = 1 - p_l \tag{8}$$

In equation 8 p_l is the possibility of performing a long transition, p_s is the possibility of performing a short transition, N_{target} is the target maximum length given to the generator. $N_{generated}$ is the number of completed transitions so far. This system is able to generate random sequences of different lengths. The length of the sequences can be controlled with N_{target} variable. Higher values of N_{target} will result in longer sequences.

Here is an example code that defines the transitions of Reber grammar:

```
    states[0].add_transition(Transition(states[1], "B"))
    states[1].add_transition(Transition(states[2], "T"))
    states[1].add_transition(Transition(states[3], "P"))
    states[2].add_transition(Transition(states[2], "S"),
    ↪  long_t=True)
    states[2].add_transition(Transition(states[4], "X"))
    states[3].add_transition(Transition(states[3], "T"),
    ↪  long_t=True)
    states[3].add_transition(Transition(states[5], "V"))
    states[4].add_transition(Transition(states[6], "S"))
    states[4].add_transition(Transition(states[3], "X"),
    ↪  long_t=True)
    states[5].add_transition(Transition(states[6], "V"))
    states[5].add_transition(Transition(states[4], "P"),
    ↪  long_t=True)
    states[6].add_transition(Transition(states[7], "E",
    ↪  end=True))

```

In this code log_t parameter marks the definition as a long transition that potentially prolongs the length of generated sequence.

3.4.2.2 Generating Supra-Regular Grammars

The state machine implementation can perform state machine transitions and randomly generate sequences of regular grammars.

For grammars beyond regular grammar like context-free and context-sensitive grammars, a memory mechanism is required to process and generate sequences. Since both A_nB_n and mirror grammars are supra-regular grammars, a push down automata mechanism with a stack memory is required. To provide this the state machine implementation is improved to support a stack mechanism.

As a result each of the transitions optionally have add or remove instructions. Add instructions adds an element on top of the stack if the transition is performed.

Here is an example on how the state transitions for the Mirror Grammar is defined

```
    states[0].add_transition(Transition(states[1], "S"))
    states[1].add_transition(Transition(states[2], "M"))
    states[2].add_transition(Transition(states[3], "E",
    ↪  end=True))
    for c in "ABCD":
        states[1].add_transition(Transition(states[1], c,
        ↪  add=c), long_t=True)
        states[2].add_transition(Transition(states[2], c,
        ↪  remove=c))
```

It's worth noting that some of the grammars (A_nB_n mirror) are more deterministic in nature and it is easier to generate a sequence with a determined length exactly. And grammars like Reber grammar (state machine) are more probabilistic in nature, so it was not directly possible to generate sequences with determined lengths exactly. While the maximum length is always checked for Reber grammar, the length of the generated sequence is correlated with the indented sequence length but since it is probabilistic, generating longer sequences has a lower probability.

3.5 Model Training

3.5.1 Training Hardware

The experiments were performed on a server with following specs:

```
OS: Debian GNU/Linux 12 (bookworm) x86_64
Kernel: 6.1.0-13-amd64
CPU: 13th Gen Intel i7-13700 (24) @ 5.100GHz
GPU: NVIDIA GeForce RTX 4090
Memory: 64067MiB
```

3.5.2 Curriculum Learning

In this study, we employ a curriculum learning strategy to manage the complexity of the training process. This strategy is predicated on the notion that the complexity of a task is closely related to the length of the input sequences.

We consider longer sequences as more complex because of their inherent requirement for longer dependency tracking. Conversely, shorter sequences are deemed less complex, as they involve shorter dependencies and are generally easier for neural networks to learn and generalize from.

The curriculum learning approach introduces training sequences of increasing length over time. This method allows the model to initially learn from simpler, shorter sequences and progressively adapt to more challenging, longer sequences.

The complexity of the sequences, and thereby the training difficulty, is controlled based on the iteration number. Early iterations involve shorter sequences, gradually moving to longer sequences in later iterations. This incremental approach is designed to optimize the learning process, enabling the model to develop a more robust understanding of the underlying patterns and dependencies.

By leveraging curriculum learning, we aim to facilitate more effective training of the neural network models, particularly in tasks where understanding long-term dependencies is crucial. This methodical increase in complexity is expected to improve the models' ability to learn and generalize from the training data, thereby enhancing their performance on complex sequence learning tasks.

3.5.2.1 No Curriculum

This curriculum is the default setting and uses no curriculum learning approach. The length of the sequence is determined by the maximum sequence length. Sequence length and iteration number relation can be seen in Figure 18.

3.5.2.2 Uniform Deterministic Curriculum

The length of the sequence starts from minimum sequence length to maximum sequence length and starts over. This provides increasing complexity as the training goes, but returns to the simpler sequences. Sequence length and iteration number relation can be seen in Figure 19.

3.5.2.3 Uniform Random Curriculum

In the Uniform Random Curriculum, the sequence length for each training instance is chosen randomly within a specified range between the minimum and maximum sequence lengths. This approach introduces variability and unpredictability in the sequence lengths, which can potentially enhance the model's ability to generalize across different lengths. Sequence length and iteration number relation can be seen in Figure 20.

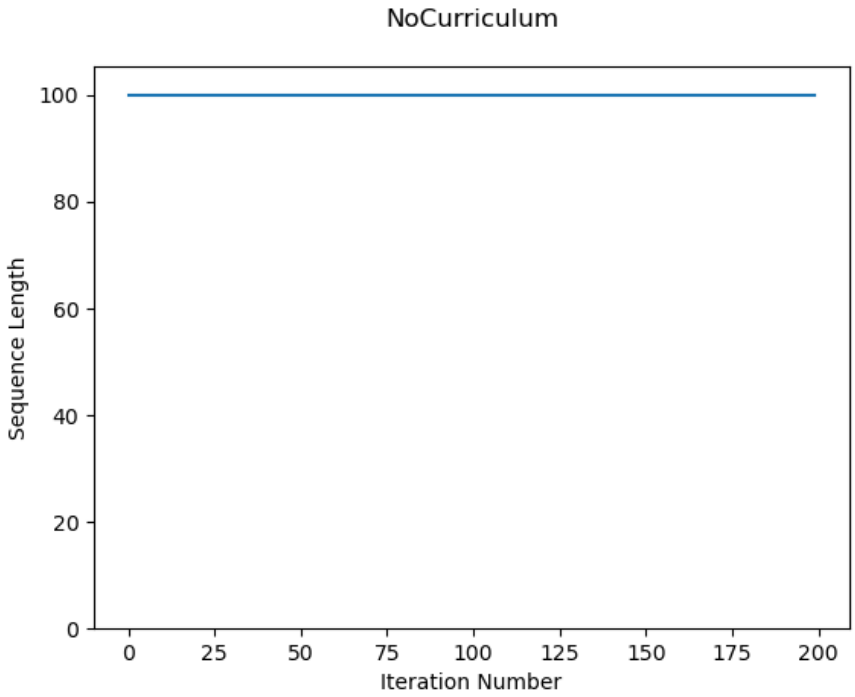


Figure 18: Graph of a sequence length with no curriculum

3.5.2.4 Gradual Increase Curriculum

The Gradual Increase Curriculum systematically increases the length of the sequences over time. Starting from the minimum sequence length, it gradually increases to the maximum sequence length throughout the training process. This method provides a more structured and steady increase in complexity compared to the random approach. Sequence length and iteration number relation can be seen in Figure 21.

3.5.2.5 Gradual Increase Spiking Curriculum

In the Gradual Increase Spiking Curriculum, the sequence length increases gradually like in the Gradual Increase Curriculum, but it occasionally introduces spikes in length. These spikes temporarily increase the sequence length to the maximum before returning to the current gradual length. This approach aims to intermittently challenge the model with more complex sequences during the gradual learning process. Sequence length and iteration number relation can be seen in Figure 22.

3.5.2.6 Simulation Experiment Generation Infrastructure

The generation and execution of experiments in this study have been meticulously designed to ensure efficiency, replicability, and systematic analysis. Each experiment involves the training of a specific

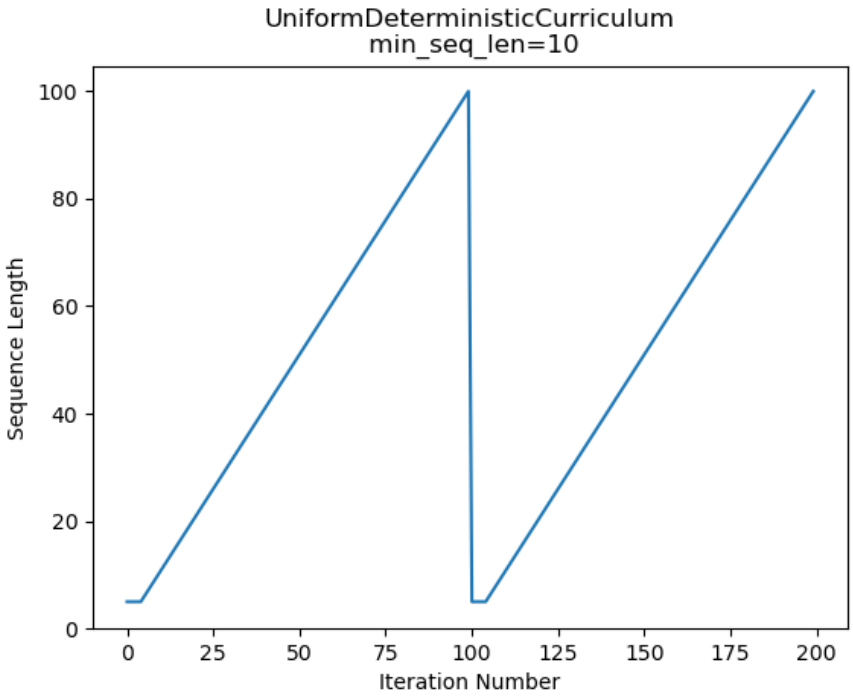


Figure 19: Graph of a sequence length with Uniform Deterministic Curriculum

neural network model on a designated task, followed by the evaluation of the model using carefully curated evaluation sets. This section outlines the infrastructure developed to facilitate these experiments.

The initiation of each experiment is versatile, allowing for both command line initiation and through a structured JSON experiment file. This dual-mode of operation offers flexibility in terms of experiment setup and execution. Upon commencement, each experiment is assigned a unique identifier, comprising the task name, model type, and a timestamp, ensuring a systematic and organized tracking system.

A critical component of the experimental infrastructure is the comprehensive logging and saving of various elements. For each experiment, the model parameters, results, the actual trained model file, and the sequences used for training are meticulously saved. This includes a detailed JSON file capturing the loss and cost for each sequence in the evaluation set, providing a granular view of the model's performance.

To enhance the robustness and diversity of the experiments, a program has been developed to randomly generate experiments. This program selects combinations from a predefined set of parameters, fostering an extensive exploration of model behaviors under different conditions. Furthermore, manual creation of experiments is facilitated through the crafting of a JSON file, offering tailored experimentation for specific research queries.

Efficiency and resource optimization are key considerations in this infrastructure. To this end, previous experiments are cached, and experiments with identical parameters and seed are not redundantly

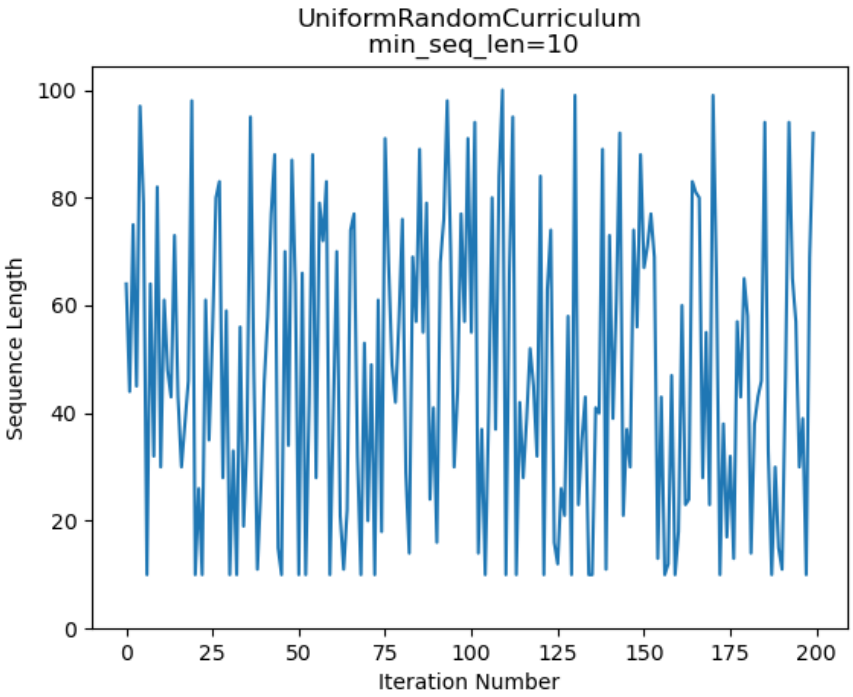


Figure 20: Graph of sequence length with Uniform Random Curriculum

generated. This approach not only saves computational resources but also ensures consistency across repeated experiments.

Finally, the infrastructure includes a program that continuously monitors a designated directory for new experiment JSON files. Upon detection, this program automatically initiates the training process. This feature is particularly useful for queueing multiple experiments, facilitating uninterrupted research workflow, and efficient utilization of computational resources.

In summary, the experiment generation infrastructure developed for this study is characterized by its flexibility, efficiency, and comprehensive data management. It supports a wide range of experimental scenarios, from highly controlled, manually defined setups to randomized, large-scale explorations, all while ensuring the integrity and reproducibility of the research.

3.5.3 Parameter Optimization

In this study, an approach to parameter optimization has been implemented with the aim of exploring effective combinations of parameters that might facilitate the generalization of sequences. This exploration is guided by the principle of examining various possibilities rather than asserting definitive optimal settings.

The method chosen for this exploration is the ‘gp_optimize‘ function from the skopt library, a tool recognized in the machine learning community for its utility in parameter tuning [173]. This function is based on Bayesian optimization using Gaussian Processes, an approach known for its potential in

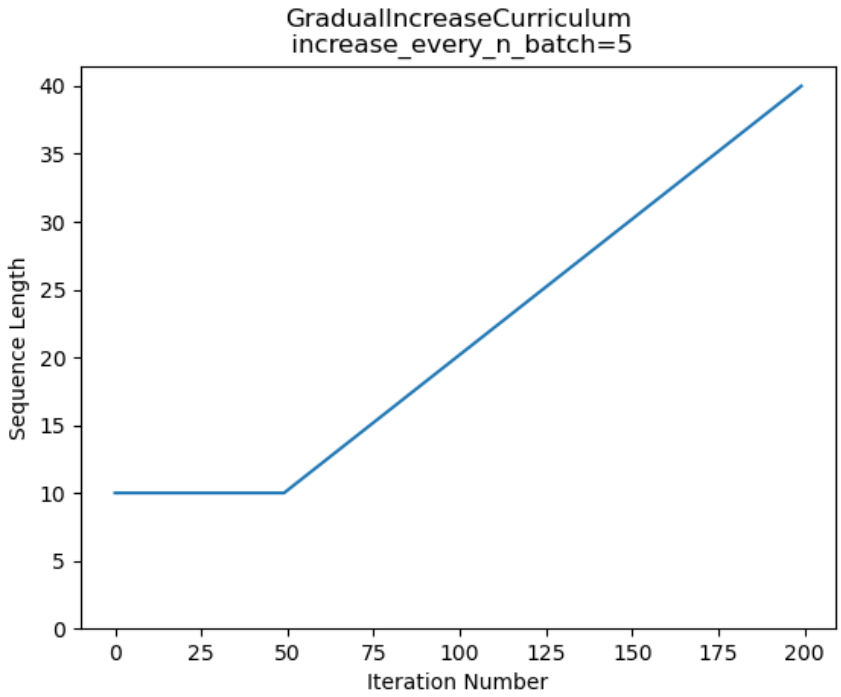


Figure 21: Graph of sequence length with Gradual Increase Curriculum

navigating complex parameter spaces in a systematic manner [174]. However, it is important to note that this method provides a guided exploration rather than definitive answers.

The bounds for the parameters in this study were set with the intention of covering a broad range, allowing for a diverse exploration of the parameter space. These bounds are not fixed but serve as a starting point for the optimization process:

- `ntm_memory_m`: Set between 5 and 100, this range explores various dimensions of NTM's memory.
- `ntm_memory_n`: Set between 50 and 1500, this explores different sizes of the NTM's memory.
- `sequence_max_len`: With a range from 10 to 100, this examines the impact of different sequence lengths.
- `rmsprop_lr`: Ranging from 0.00001 to 0.001, this explores learning rates in the RMSprop optimizer.
- `rmsprop_momentum`: Set between 0.0 and 0.9 to observe the effects of varying momentum in the optimizer.
- `rmsprop_alpha`: Ranging from 0.1 to 0.99, this parameter explores different smoothing constants.
- `spike_modifier`: Set between 0.1 and 2.0, to understand the influence of spike modification.

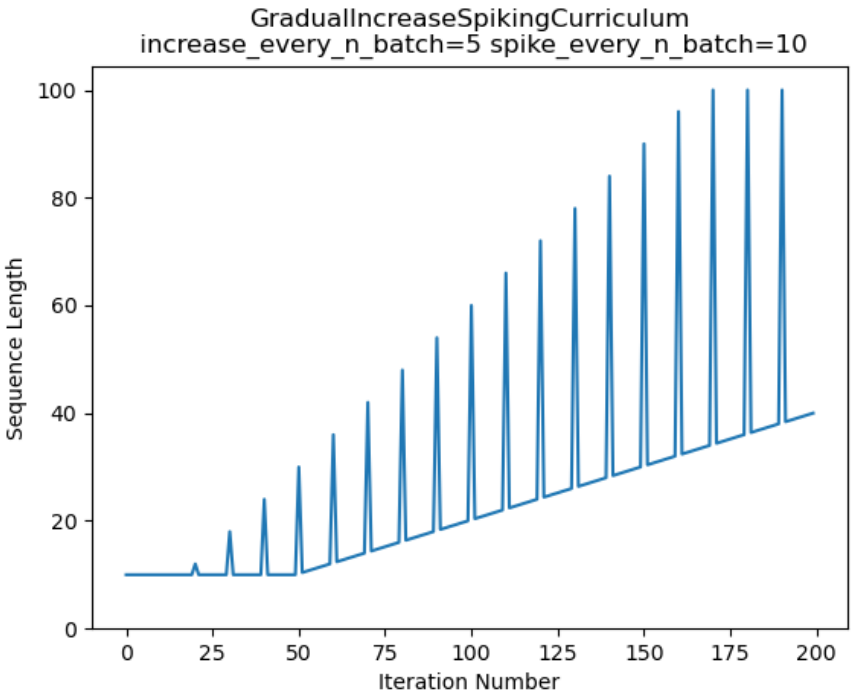


Figure 22: Graph of sequence length with Gradual Increase Spiking Curriculum

The parameter search tool, while systematic in its approach, is primarily exploratory in nature. It navigates through the test space within these bounds, allowing for the examination of a variety of parameter combinations. The application of Bayesian optimization provides a structured yet exploratory framework for this investigation, helping to highlight potential areas of interest in the parameter space.

3.6 Output Evaluation

3.6.1 Metrics for Assessing Model Performance

3.6.1.1 Calculation of the Cost Metric

In this study, we evaluate the performance of our models using a 'cost' metric. The computation of this metric is an integral part of assessing the effectiveness of the models in predicting possible state transitions accurately. The cost metric is calculated through a series of steps that transform the output probabilities into a quantifiable measure of prediction accuracy.

Firstly, at each step of the model's output, we extract the output vector, which represents the probabilities of possible transitions. Each element in this vector is a probability value ranging between 0 and 1. These probabilities are then subjected to a thresholding process, where values are binarized based on a threshold of 0.5. This results in a binary vector consisting of 0's and 1's, representing the model's predictions in a simplified, binary form.

Subsequently, this binarized output vector is compared against the correct labels, which are the actual possible transitions at each stage of the sequence. The comparison involves checking each element of the binarized output vector against its corresponding correct label. For every element where the prediction (binarized output) differs from the actual label, the cost is incremented by 1.

This method of calculating the cost thus provides a clear and quantifiable measure of the model's performance. It reflects the number of inaccuracies in the model's predictions by quantifying the falsely predicted state transitions. The lower the cost, the more accurate the model is in predicting the possible transitions, indicating a higher level of performance in sequence prediction tasks.

3.6.2 Evaluating Generalization and Long-Term Dependency

The evaluation of generalization performance and long-term dependency in neural network models is a critical aspect of this study. Our approach to assessing generalization hinges on the models' ability to handle sequences that are significantly longer than those they were trained on. This methodology ensures that the models are not merely memorizing sequences but are genuinely learning and applying the rules of the grammar to novel situations.

For a precise and standardized assessment, we utilize the cost metric outlined in Section 3.6.1.1. This metric serves as a quantitative measure of a model's performance, especially in terms of its capacity to generalize beyond the training data. The cost metric is calculated for each model, providing a consistent basis for comparison across different architectures.

A critical aspect of our evaluation strategy is the length of the sequences used in the test set. By including sequences that are considerably longer than the training sequences, we introduce a challenging scenario that tests the models' ability to generalize. This approach is particularly effective in differentiating models that have truly understood the underlying structure of the grammar from those that have merely memorized the training sequences.

The ultimate indicator of a model's success in this test is a cost metric of 0. Achieving this score implies that the model has successfully predicted all possible outcomes at each stage of every sequence in the test set. Such an outcome would strongly suggest that the model has not only memorized the sequences but has also internalized the rules of the grammar to a degree that allows for accurate predictions in novel and more complex scenarios. This level of performance would be indicative of a profound understanding of the grammar, reflecting a significant achievement in the realm of neural network-based cognitive modeling.

In summary, our approach to evaluating generalization and long-term dependency is designed to rigorously test the models' ability to apply learned grammatical rules to novel and more complex scenarios. This is achieved through the use of a standardized cost metric and the inclusion of longer sequences in the test set, ensuring that the evaluation process is both challenging and indicative of the models' true learning capabilities.

3.7 Output Interpretation

3.7.1 Human-Readable Output

To be able to evaluate each individual output we developed a functionality to visualise expected and given outputs.

An example of the Reber grammar that is predicted by a generalized NTM model is shown in Table 3.

Table 3: Inputs, target and given possible outputs for a Reber grammar

Iter	Input	Target	Output
1	B	TP	TP
2	T	SX	SX
3	S	SX	SX
4	S	SX	SX
5	X	SX	SX
6	X	TV	TV
7	T	TV	TV
8	T	TV	TV
9	T	TV	TV
10	V	VP	VP
11	P	SX	SX
12	S	E	E
13	E	-	-

We also have the feature to see only the mismatching outputs shown in Table 4.

Table 4: Inputs and mismatching target and possible outputs for a Reber grammar

Iter	Input	Target	Output
2	T	SX	T
9	T	TV	VE
10	V	VP	VE
11	P	SX	VE
12	S	E	VE
13	E	-	VE

3.7.2 Visualization Strategies

To effectively interpret the outputs and internal mechanisms of neural networks, particularly NTMs, we have implemented various visualization tools. These tools are helpful for understanding how the

models process and respond to different inputs. The subsections below detail these visualization strategies.

3.7.2.1 Output Visualization as Sequence

One useful approach is visualizing the outputs of a model for a given sequence alongside the target outputs (expected outputs). This method, as depicted in Figure 23, allows for a direct comparison between what the model predicts and what it is supposed to predict. This visualization can be particularly insightful in assessing the accuracy and efficiency of a model in real-time processing of sequences.

Additionally, we can visualize the predicted probabilities for each output, as illustrated in Figure 24. In this visualization, the intensity of the color in the boxes correlates with the probability values; darker boxes indicate probabilities closer to 1. This method provides a more granular view of the model's confidence in its predictions, offering deeper insights into its decision-making process. By examining these probability distributions, we can better understand the model's certainty in various stages of sequence processing and how it aligns with the expected outcomes.

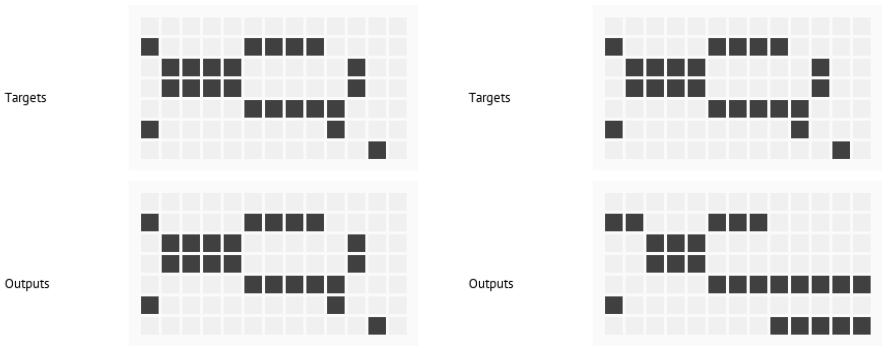


Figure 23: Visualisation of target values and outputs of generalized (left) and non-generalized (right) instances for Reber grammar. The model begins to give the wrong outputs after the 8th timestep.

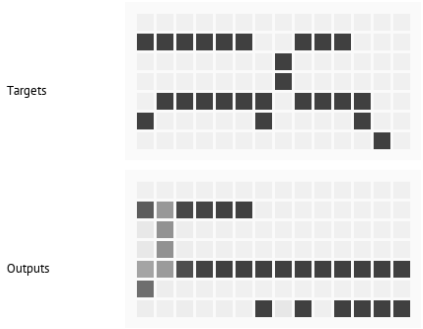


Figure 24: Visualisation of target value probabilities for Reber grammar. The probabilities for tokens are greyed out meaning outputs are lower than 1 but higher than 0 for those tokens.

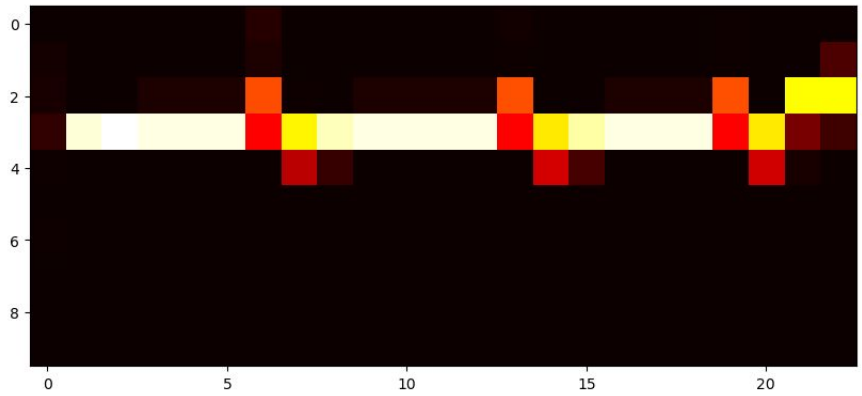


Figure 25: Example heatmap visualization of an NTM read head processing a Reber grammar sequence. X axis is the iteration number and Y axis is the memory address. Dark cells represent lower values, bright cells represent higher values.

3.7.2.2 Memory Access Visualization

Memory access visualization offers a window into the temporal dynamics of how models like NTMs interact with their memory during input processing. By mapping out the memory read and write operations over time, we can observe the model's strategy for accessing and utilizing information stored in memory.

The provided example of a NTM read head graph, Figure 25, serves as an illustrative case of such visualization. In this heatmap, the x-axis represents the time steps during the processing of a sequence, and the y-axis corresponds to the individual memory locations. The color intensity at each point in the graph reflects the degree to which a particular memory location is read at a given time step; the brighter the point, the higher the read intensity.

From this graph, we can deduce several facets of the model's behavior:

Temporal Patterns: We can identify specific moments when the model preferentially reads from certain memory locations, which may correspond to critical points in the sequence processing where specific information is required.

Memory Utilization: The spread of color across the memory locations might indicate the extent of memory utilization. A focused band of intense color would suggest that the model is relying on a smaller segment of memory, while a more distributed pattern would imply wider memory usage.

Model Efficiency: Sparse use of intense colors might suggest that the model is efficiently recalling information as needed, rather than constantly reading from memory, which could be indicative of a well-optimized memory access strategy.

Insight into Learning: Repeated patterns or consistent access strategies across similar sequences can hint at the model's learning and generalization capabilities, showing us how it abstracts and applies the grammatical rules it has learned.

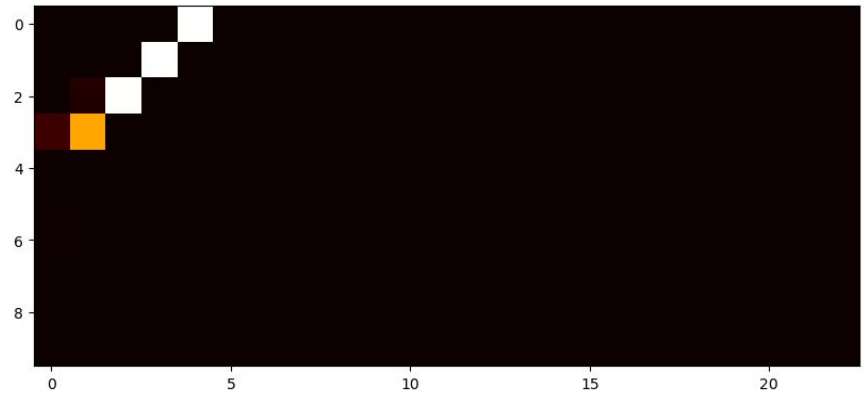


Figure 26: Example heatmap visualization of an NTM write head processing a Reber grammar sequence. X axis is the iteration number and Y axis is the memory address. Dark cells represent lower values, bright cells represent higher values.

If Figure 26 we can see an example of NTM read head graph. Both having access to read and write head graphs can help analysing what kind of information might be written to memory and read from memory at any point in the sequence.

By examining such visualizations, we can gain insights into the model's internal mechanism and potentially identify areas for optimization or further investigation. These visualizations, while not mandatory, can significantly aid in the interpretative analysis of complex neural network behaviors.

3.8 Summary and Transition to Results

In this chapter, we have systematically delineated the methodologies underpinning our exploration of memory networks within neurocomputational models, particularly emphasizing Neural Turing Machines (NTMs), Long Short-Term Memory (LSTM) networks, and Legendre Memory Units (LMUs). Our investigation has been centered around the learning, processing, and generalization capabilities of these networks in the context of various artificial grammars, namely State Machine (Reber grammar), $\mathrm{A}_n\mathrm{B}_n$ Grammar, and Mirror Grammar.

The experimental approach, characterized by its methodical structure, was carefully crafted to probe the intricate dynamics of memory mechanisms and sequence learning in neural networks. This was achieved through a combination of procedural grammar generation, rigorous model training, and comprehensive output analysis. The choice of these specific grammars provided a consistent and controlled framework, enabling an in-depth evaluation of each model's performance against the key cognitive modeling parameters.

Key aspects of our analysis included assessing the models' proficiency in predicting novel sequences, their memory recall efficacy, and the extent of their generalization capabilities. To enhance the clarity and interpretability of our findings, a range of visualization tools and detailed parameter analysis were employed, thus ensuring a robust and transparent evaluation process.

In the following section we will look at the interpretation of the empirical data obtained from these experiments. The subsequent chapters will articulate a discussion of each model's performance, offering insights into their relative strengths and limitations in simulating cognitive functions. This analysis will not only address their capabilities in handling complex grammatical structures but will also delve into the nuances of their memory utilization and long-term dependency management. The forthcoming results are anticipated to contribute substantially to the domain of computational cognitive models, bridging the gap between theoretical constructs in cognitive science and their practical realization in neural network architectures.

CHAPTER 4

RESULTS

In this chapter, we will present the results obtained during simulation experiments conducted with LSTM, LMU, and NTM models on Reber, A_nB_n , and Mirror grammars. These experiments focus on understanding the generalization capabilities of different neural network architectures on different tasks and also shed light on their inner working mechanisms.

In the next chapter, we will present the results obtained for the mildly context sensitive grammar task, which builds on the results of this chapter.

During each simulation experiment a model was trained on a specific grammar task with sequences that have a maximum length of 100. Following training, each model was tested with a test dataset prepared for each grammar task including sequences with lengths higher than 100. This way, we aimed to test if the model could generalize to larger sequences that were out of the distribution of training data, which is considered as evidence of generalized learning of the corresponding grammar.

The following sections will provide results for different model and grammar combinations. The results will be given as the cost metric defined in Section 3.6.1.1.

4.1 Methodology for Evaluating Results

Our analysis primarily focuses on the cost metric, with particular emphasis on the lowest cost achieved in each experimental scenario. The cost metric, as detailed in Section 3.6.1.1, quantifies the model's accuracy in predicting possible state transitions. A lower cost indicates better performance, with zero cost representing perfect prediction.

The rationale behind examining the lowest cost is to investigate the generalization capability of the architectures rather than the frequency of successful generalization. This approach aligns with our interest in competence (the ability to learn) over performance (the ease or frequency of learning). As explained in Section 3.3.8, our evaluation framework employs a stringent comparison method. Consequently, when a specific model achieves a cost of zero, it strongly indicates that the architecture has successfully learned the underlying rules of the grammar from the provided examples.

For instance, consider two hypothetical scenarios:

- Scenario A: Model X achieves costs of [0, 50, 100] across three trials

- Scenario B: Model Y achieves costs of [10, 20, 30] across three trials

In this case, we would consider Model X to have demonstrated superior competence, despite its less consistent performance, as it managed to achieve perfect prediction in at least one instance.

While our primary focus is on the lowest cost, we also consider the mean and standard deviation of the cost metrics to provide insights into the ease and consistency of generalization. These secondary analyses offer a more comprehensive view of each architecture's performance characteristics.

It is important to note that this evaluation approach prioritizes the potential capabilities of the architectures over their average performance. This aligns with our research objective of understanding the computational properties that enable human-like grammar learning. However, we acknowledge that in practical applications, consistency of performance might be equally crucial.

In the following sections, we will present the results of our experiments, interpreting them through the lens of this evaluation methodology. We will examine how different architectures perform across various grammar types, with a particular focus on instances where perfect generalization (zero cost) is achieved.

4.2 Evaluation Set

The trained models are evaluated on previously recorded datasets consisting of sequences of varying lengths. The main idea behind the training sets is that they should contain sequences much larger than the original sequence length that the models were trained on which was limited to 100. Then the cost for the dataset will be calculated on those evaluation datasets. Properties of these datasets can be seen in Table 5.

Table 5: The statistics of the evaluation set for various grammar types

Grammar Type	Min Seq. Len.	Max Seq. Len.	Mean Seq. Len.	# of Sequences
Reber	21	185	88.80	5
A_nB_n	100	800	400.00	4
Mirror (WW^R)	49	499	211.50	4

4.3 Overall Results

The minimum cost values for different combinations of Model Types and Tasks are presented in Table 6. To assess the results, we will examine the cost for each model as described in Section 3.6.1.1. Additionally, we will evaluate their generalization performance by determining whether the cost reaches zero, as explained in Section 3.6.2.

These results show that all of the grammars in the study were able to generalize on NTM. LSTM was able to generalize the Reber grammar and the A_nB_n grammar. LMU was able to generalize for the Reber grammar and almost generalize for the A_nB_n grammar with a few exceptions.

Results also suggest that for Reber grammar LSTM architecture performed much more consistently, as indicated by low "cost", "average" and "standard deviation" values.

Table 6: The cost for model and task combinations

		Cost			
		min	max	mean	std
Grammar Type	Model Type				
A_nB_n Grammar	LMU	2.25**	893.0	478.45	180.96
	LSTM	**0***	1117.25	290.18	287.12
	NTM	**0***	932.75	259.84	218.35
Mirror Grammar	LMU	101.75	746.5	522.35	146.27
	LSTM	88.5	770.25	377.52	220.13
	NTM	**0***	886.5	492.52	185.55
Reber Grammar	LMU	**0***	331.2	40.42	65.52
	LSTM	**0***	112.8	11.28	35.67
	NTM	**0***	228.0	37.1	51.89

* The model-task combinations that generalized
** The model-task combinations that generalized with a few exceptions

It should be noted that for the mean cost LSTM performed the best for Mirror grammar and Reber grammar and was close to the best mean cost value for A_nB_n grammar. Especially for shorter sequences LSTM was able to produce low costs as well. But for the context of this study we will focus mostly on generalization (0 cost) rather than average performance. Specifically the models capacity to generalize, namely being able to get zero cost in at least one example will be the major factor for architecture success.

4.4 Reber Grammar (State-Machine) Results

4.4.1 LSTM Performance

Figure 27 shows the minimum cost for each combination of parameters. The size of the LSTM memory (size of hidden layer) is indicated in the y axis. As we can see in Figure 27, LSTM architecture performed remarkably well on Reber grammar regardless of the memory size (hidden size). A small number of experiments were sufficient to see that LSTM has the ability to generalize the rules of Reber grammar beyond its training set sequence lengths. This indicates that LSTM has the ability to learn the rules of a regular grammar.

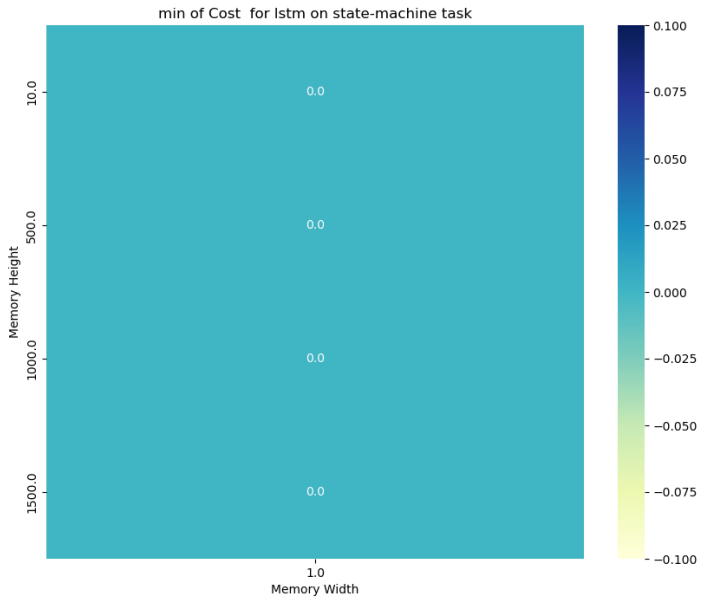


Figure 27: Cost of the best performing (lowest cost) results for LSTM on Reber grammar.

4.4.2 LMU Performance

Figure 28 shows the minimum cost for each combination of parameters. 'Memory Height' is the number of legendre polynomials and 'Memory Width' is the model layer size as explained in Section 3.3.9. As we can see in Figure 28 LMU architecture performed near perfectly on Reber grammar regardless of memory width and length. Results indicate that LMUs have the ability to generalize the rules of the Reber grammar beyond its training set sequence lengths. This indicates that LMUs have the ability to learn the rules of a regular grammar.

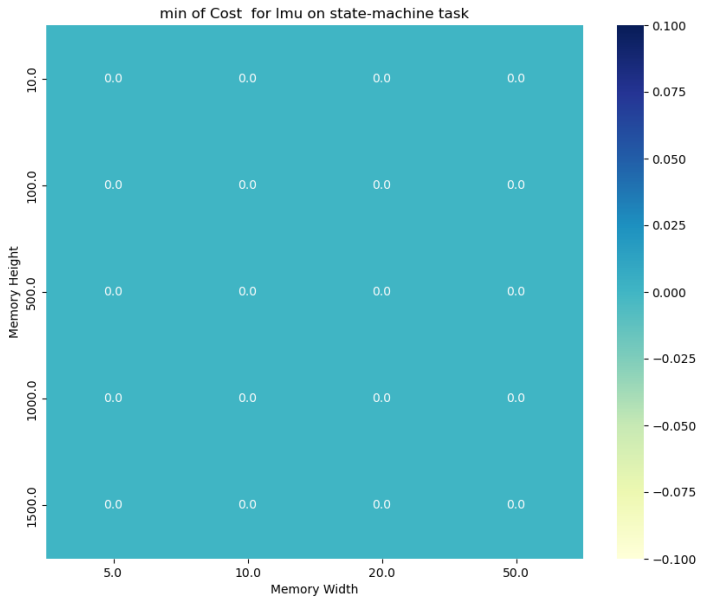


Figure 28: Cost of the best results (lowest cost) for LMU on Reber grammar.

4.4.3 NTM Performance

As we can see from the Figure 29 NTM architecture performed considerably well on Reber grammar regardless of memory width and length.

The results indicate that NTMs have the ability to generalize the rules of Reber grammar beyond its training set sequence lengths.

This indicates that NTMs have the ability to learn the rules of a regular grammar. Performance independence of memory width indicates that the recorded internal representations were small enough to fit in a memory with size of at least 5.

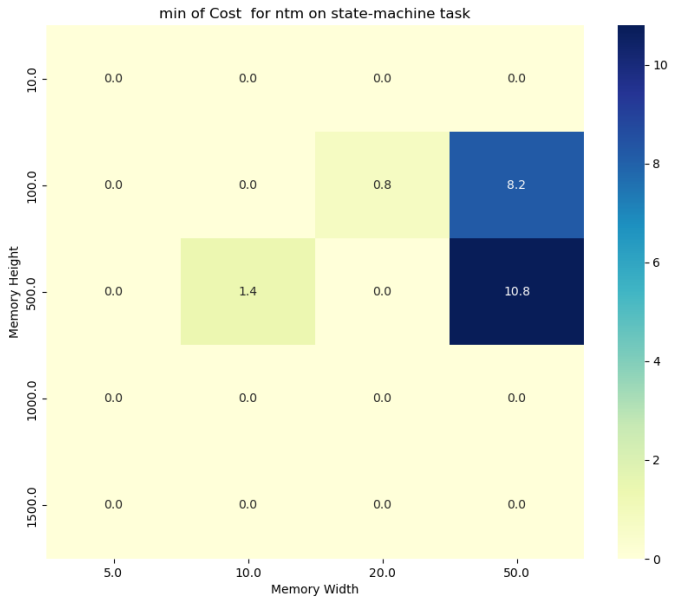


Figure 29: Cost of the best results (lowest cost) for NTM on Reber grammar.

4.4.4 Overview of Model Performances on Reber Grammar

The results of the Reber grammar experiments support the idea that regular grammar rules can be learned by networks that have the ability to effectively track states through sequences. Memory networks such as NTMs can track the state through its memory mechanism, and recurrent neural networks such as LSTMs and LMUs can accomplish this with its recurrent memory.

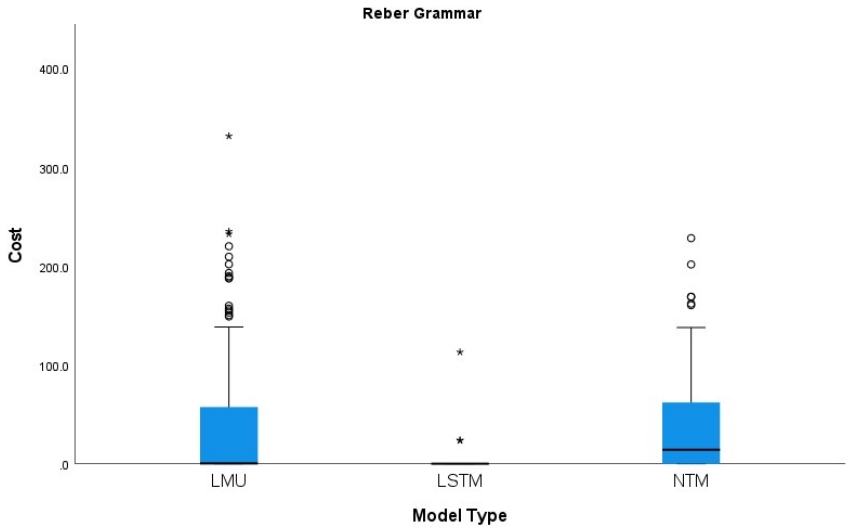


Figure 30: A boxplot showing the distribution of costs for different model types for Reber grammar

Figure 30 Shows both models were able to touch 0 meaning they all could achieve generalization. LSTM was able to generalize easily except a few outliers with almost all of the instances being able to generalize. LMU has a very low median cost. NTM is similarly distributed but with a slightly higher median cost.

4.5 A_nB_n Grammar Results

The results of the experiments training various models on data generated from A_nB_n grammar are presented in the following subsections.

4.5.1 LSTM Performance

Figure 31 shows the minimum cost for each combination of parameters. Size of the LSTM memory (size of the hidden layer) is indicated in the y axis. As we can see from the Figure 31 LSTM architecture performed near perfectly on A_nB_n grammar regardless of memory (hidden layer) size. The results indicate that the LSTMs have the ability to generalize the rules of the A_nB_n beyond its training set sequence lengths. It shows that LSTMs have the ability to learn the rules of some of the context free grammars, one of them being A_nB_n .

In some of the cases, rules were learned with few minor exceptions. Memory parameters and performance do not have a direct correlation indicating that the learned strategy did not involve remembering each of the previous tokens but rather than counting the number of the tokens.

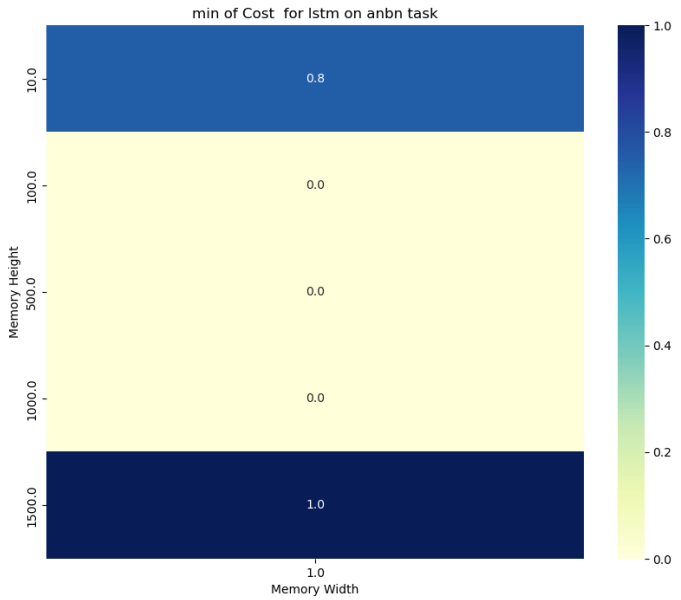


Figure 31: Cost of the best results (lowest cost) for LSTM on A_nB_n grammar.

4.5.2 LMU Performance

Figure 32 shows the minimum cost for each combination of parameters. 'Memory Height' is the number of legendre polynomials and 'Memory Width' is the model layer size as explained in Section 3.3.9.As we can see from the Figure 32 LMU architecture showed capacity to learn A_nB_n .

The instances that were close to generalization were of different memory parameters. This supports the idea that the performance and strategy is not memory-contrained.

Results indicate that LMUs have the ability to generalize most of the rules of A_nB_n beyond its training set sequence lengths. This indicates that LMUs have the ability to learn the rules of some context-sensitive grammar, one of them being A_nB_n .

The number of instances that came close to generalization are much lower than LSTM, indicating that LMU memory mechanism might not be very efficient for these kinds of tasks. LMUs were able to learn with small memory widths, suggesting that the learned strategy involves counting the number of A tokens.

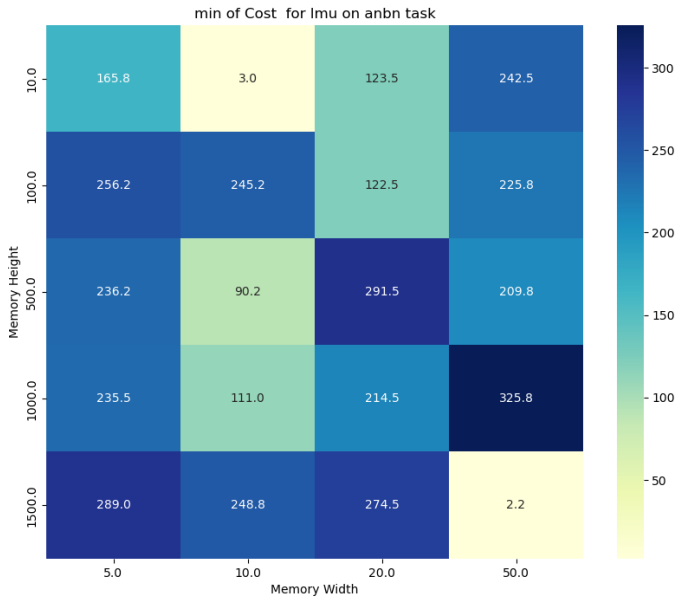


Figure 32: Cost of the best results (lowest cost) for LMU on A_nB_n grammar.

4.5.3 NTM Performance

Figure 33 shows the minimum cost for each combination of parameters.As we can see from the Figure 33 NTM architecture performed greatly in learning A_nB_n grammar.

The instances that generalized well have greater memory heights. This might indicate that a token storing strategy might be used instead of counting but there are some instances that came close to generalizing despite lower memory heights. There are no conclusive results for the strategy performed by NTM models.

Results indicate that NTMs have the ability to generalize the rules of the A_nB_n beyond its training set sequence lengths. This indicates that LSTMs have the ability to learn the rules of some context free grammar, one of them being A_nB_n .

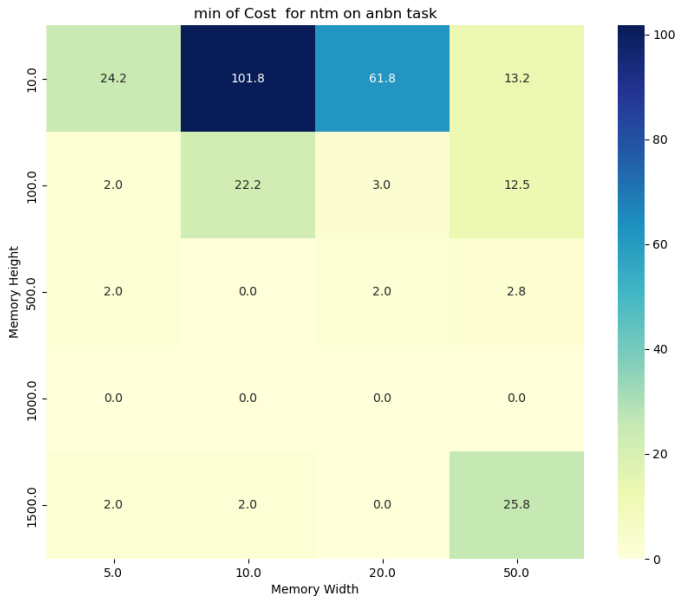


Figure 33: Cost of the best results (lowest cost) for NTM on A_nB_n grammar.

4.5.4 Overview of Model Performances on A_nB_n grammar

A_nB_n rules were fully generalized by some instances of LSTMs and NTMs. LMUs had some instances that had relatively low costs. It made some mistakes and was able to generalize most of the rules. The counting strategy allowed the generalization of the grammar with low memory constraints.

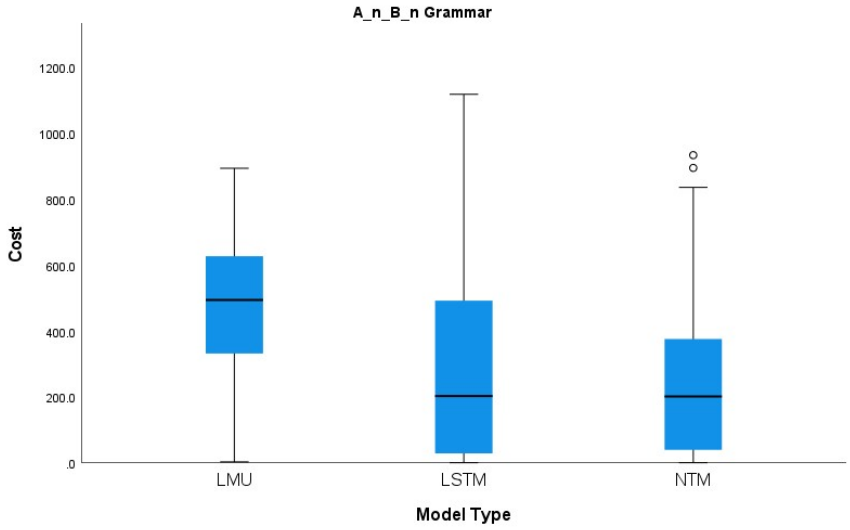


Figure 34: A boxplot showing the distribution of costs for different model types for A_nB_n grammar

Figure 34 shows each model's whiskers touching 0 cost meaning that they were able to generalize. LMU is shown to have a higher median cost than LSTM and NTM. LSTM and NTM have similar median cost but NTM is observed to have lower IQR implying slightly higher stability.

4.6 Mirror Grammar Results

The following subsections include the experiment results for various grammars and models.

4.6.1 LSTM Performance

Figure 31 shows the minimum cost for each combination of parameters. Size of the LSTM memory (size of the hidden layer) is indicated in the y axis. As we can see from the Figure 35 LSTM architecture performed poorly in learning Mirror grammar. The system was unable to come close to generalization in any of the various parameters that were tried.

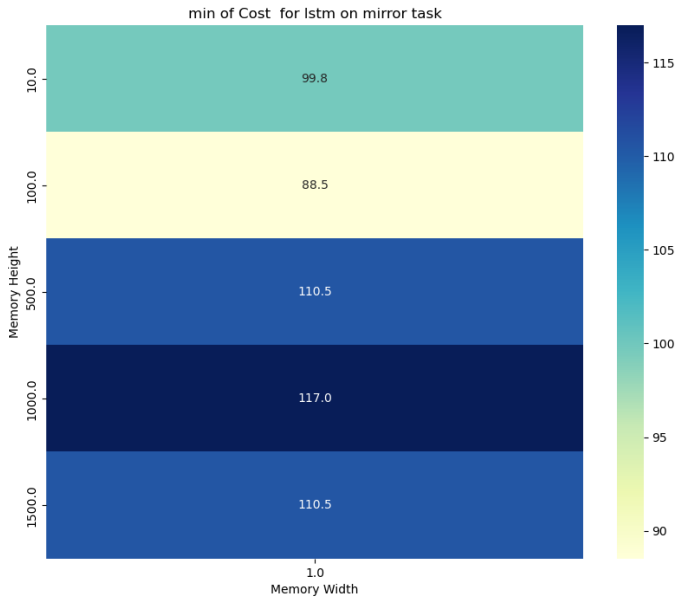


Figure 35: Cost of the best results (lowest cost) for LSTM on Mirror grammar.

4.6.2 LMU Performance

Figure 36 shows the minimum cost for each combination of parameters. 'Memory Height' is the number of legendre polynomials and 'Memory Width' is the model layer size as explained in Section 3.3.9. As we can see from the Figure 36 LMU architecture performed poorly in learning Mirror grammar.

The system could not come close to generalization in any of the various parameters that were tried.

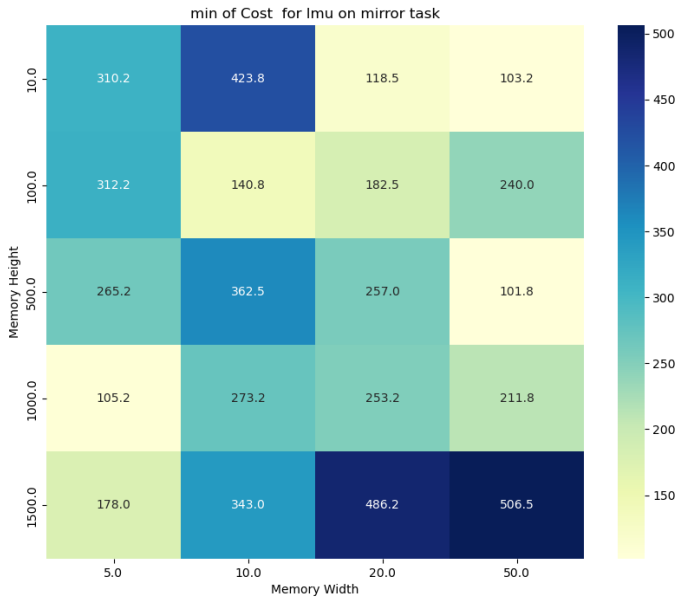


Figure 36: Cost of the best results (lowest cost) for LMU on Mirror grammar.

4.6.3 NTM Performance

Figure 37 shows the minimum cost for each combination of parameters.As we can see from the Figure 37 NTM architecture was able to learn and generalize Mirror grammar.

The NTM was only able to generalize when the memory height was 500 or more. There were instances of generalization for each memory width, indicating that memory width does not play a significant role for this problem.

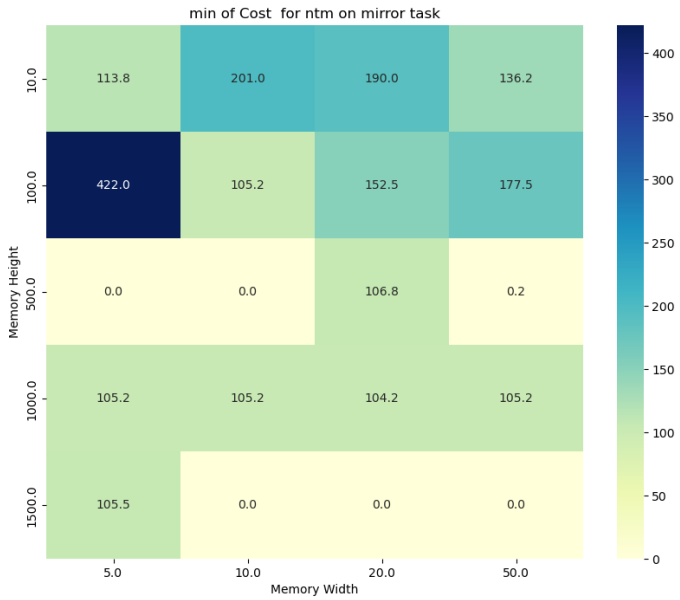


Figure 37: Cost of the best results (lowest cost) for NTM on Mirror grammar.

4.6.4 Overview of Model Performances on Mirror grammar

In the Mirror grammar tasks, RNN type neural networks (LSTM and LMU) failed to learn and generalize the rules of the grammar. The mirror grammar requires the storage and recalling of exact tokens. Each of the items should be recalled to perform the task correctly.

Mirror grammar computationally requires the equivalent mechanism of push-down-automata (PDA). In the experiments, NTMs showed the capacity to act as the stack memory of a PDA.

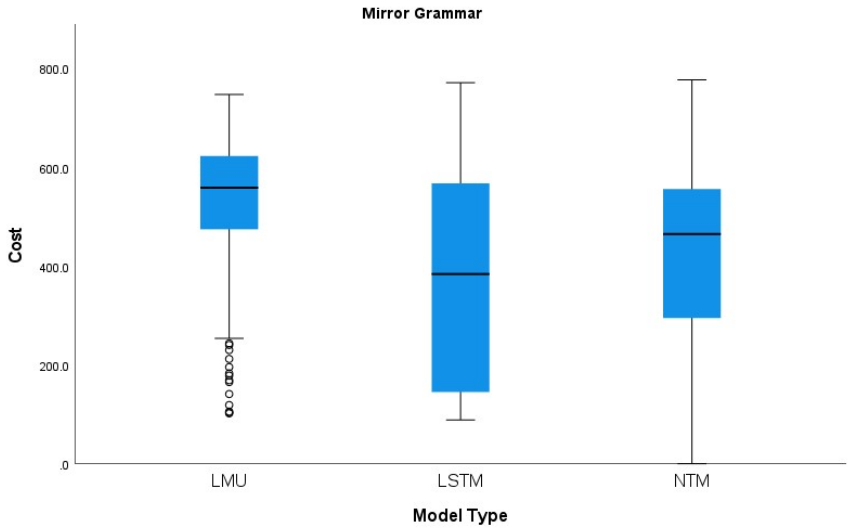


Figure 38: A boxplot showing the distribution of costs for different model types for Mirror grammar

Figure 38 shows only NTM model whiskers minus 0 cost, which means that NTM was able to generalize. However very few instances were able to reach 0 cost. LSTM has lower median cost than LMU and even NTM. This shows an interesting property of the metric where the average lower cost is not indicative of the model capacity. LSTM can show better performance in learning the statistical regularities without learning the rules.

4.7 Analysis of Generalization in LSTM, LMU, and NTM Models

This section explores the distinction between instances where various models generalized effectively and those where they did not, highlighting the notable behavioral divergences between these instances.

In the context of the Reber grammar, LSTM models failing to generalize often exhibited outputs characterized by uncertainty and ambiguity, as illustrated in Figure 39. These outputs lacked precision, suggesting a vague comprehension of potential transitions.

For the A_nB_n grammar, LSTM models that failed to generalize typically struggled with accurately counting the 'B' tokens, often terminating sequences prematurely. This pattern is depicted in Figure 40.

Regarding Legendre Memory Units (LMUs), non-generalizing models for Reber grammar often showed confusion at the beginning and end of sequences. This phenomenon, shown in Figure 41, might stem from the unique occurrences of Begin and End tokens in each sequence, posing challenges to LMUs' memory mechanisms.

For the A_nB_n grammar, LMUs that nearly generalized but fell short appeared to recognize the transition from 'A' to 'B'. However, they often failed to maintain an accurate count of 'A's, resulting in excessive 'B' outputs, as depicted in Figure 42.

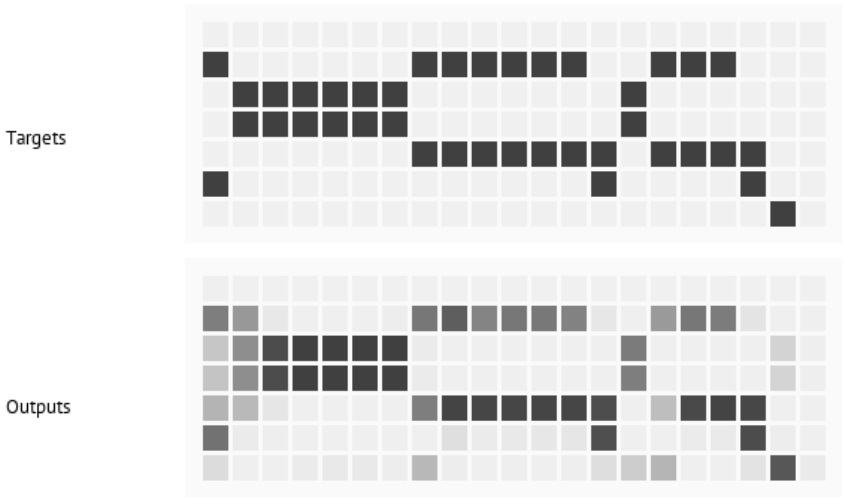


Figure 39: Comparison of outputs and target outputs in a non-generalizing LSTM instance for Reber grammar

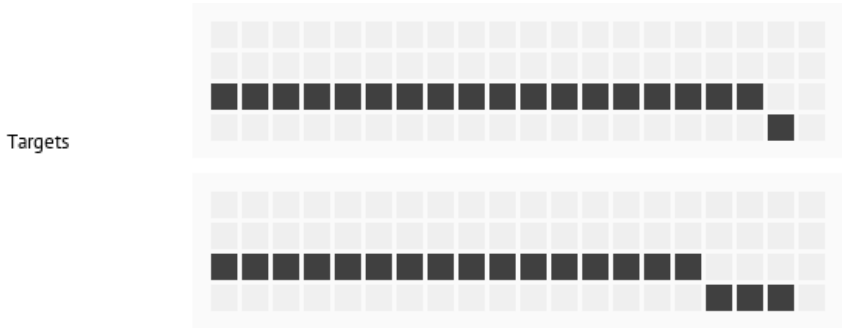


Figure 40: Last 20 outputs versus target outputs in a non-generalizing LSTM instance for A_nB_n grammar

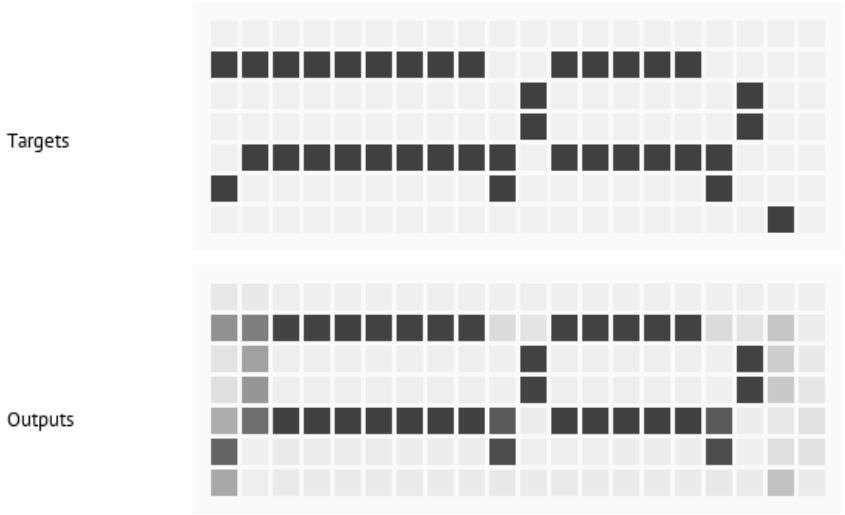


Figure 41: Outputs versus target outputs in a non-generalizing LMU instance for Reber grammar

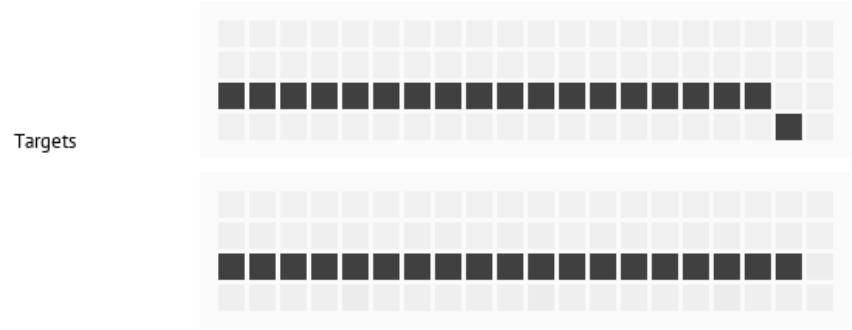


Figure 42: Last 20 outputs versus target outputs in a non-generalizing LMU instance for A_nB_n grammar

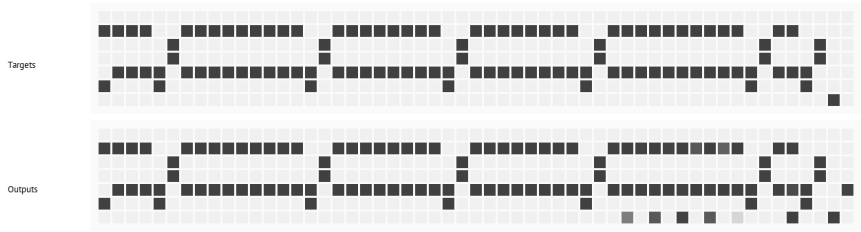


Figure 43: Outputs versus target outputs in a non-generalizing NTM instance for Reber grammar

Neural Turing Machines (NTMs) exhibited a notable pattern in non-generalizing instances with Reber grammar. Over time, the output quality degraded, and models often incorrectly identified the end state, as shown in Figure 43. This issue might be attributed to inadequate learning of certain states.

For the A_nB_n grammar, NTMs displayed a behavioral profile similar to LMUs in scenarios where they failed to generalize, as illustrated in Figure 44.

In summary, the analysis reveals distinct patterns in LSTM, LMU, and NTM models' handling of Reber and A_nB_n grammars, particularly in instances of failed generalization. These patterns offer insights into the strengths and limitations of each model type in processing complex grammatical structures.

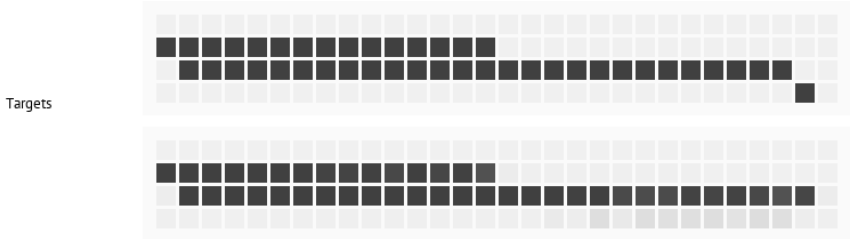


Figure 44: Outputs versus target outputs in a non-generalizing NTM instance for A_nB_n grammar

4.7.1 NTM Memory Access

The memory access mechanism of the NTM can be tracked for each part of a sequence during an inference of the neural network. Correctly generalized instances show read or write access patterns that are parallel to the state or memory to be represented at that point of time. For example Figure 48 shows that this network uses the first sequences as a chance to write tokens to different addresses. Figure 45 shows that it performs reads in parallel to the state the state machine is in.

Similarly, Figure 49 shows the read patterns of A_nB_n grammar. Once the A's are completed, the system begins to change the read addresses to understand what the possible next tokens are. This graph shows that this particular instance solved the A_nB_n grammar problem not just by counting the number of tokens directly but rather it reads previously written representations when in a correct state.

Another example of a correctly generalized NTM instance is for the Mirror grammar. As seen in Figure 46 accessed memory addresses increase one the sequence reaches the middle point. Then begins to recall the previously written sequences, correctly and flexibly mirroring the previous tokens without any error.

As opposed to those we can see that the instances that failed to generalize do not have the logical memory access patterns. For example Figure 47 shows the write pattern of a NTM when trying to predict next tokens of Reber grammar. The memory write operations converge to a single address. As we can see in Figure 50 the read patterns do not converge to a single address. The system has failed to learn to read and write memory in a temporally meaningful way.

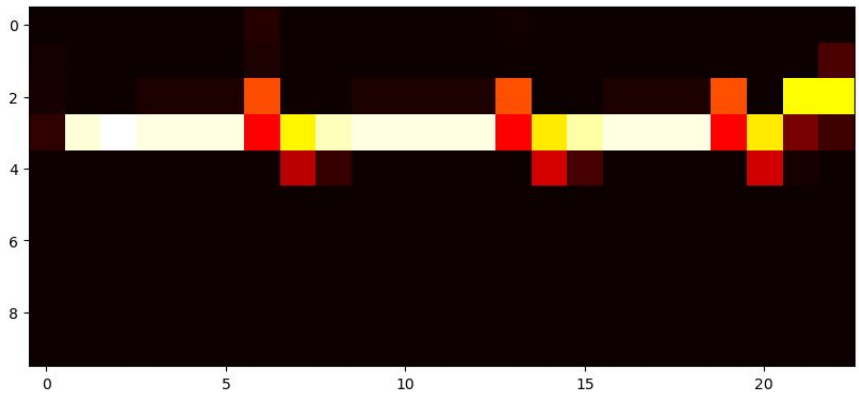


Figure 45: NTM memory read access for a correctly generalized instance of Reber grammar. X axis is the sequence index and Y axis is the memory address.

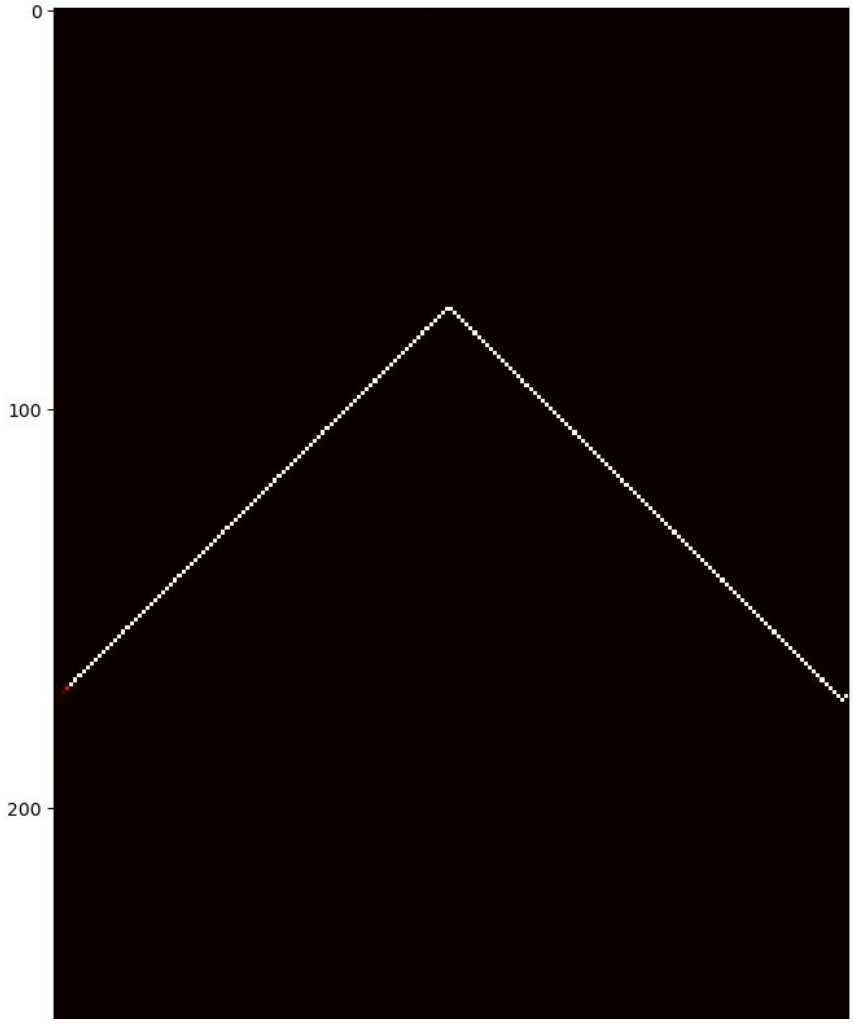


Figure 46: NTM memory read access for a correctly generalized instance of Mirror grammar. X axis is the sequence index and Y axis is the memory address.

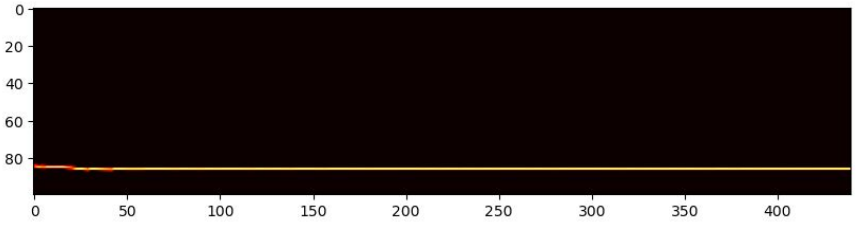


Figure 47: NTM memory write access for an incorrectly generalized instance of Reber grammar. X axis is the sequence index and Y axis is the memory address.

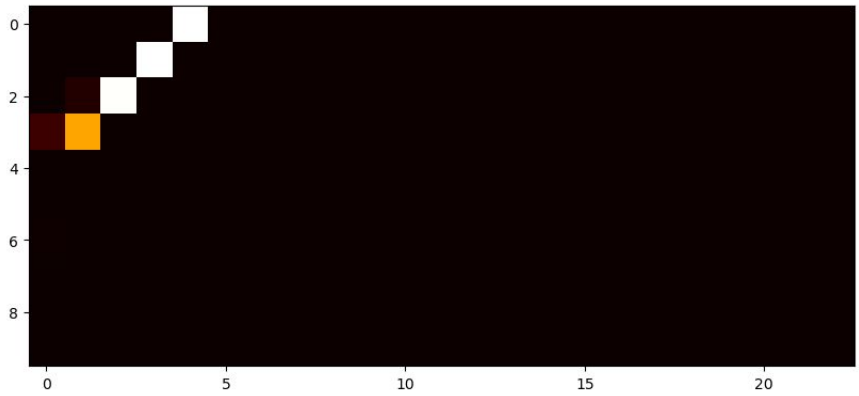


Figure 48: NTM memory write access for a correctly generalized instance of Reber grammar. X axis is the sequence index and Y axis is the memory address.

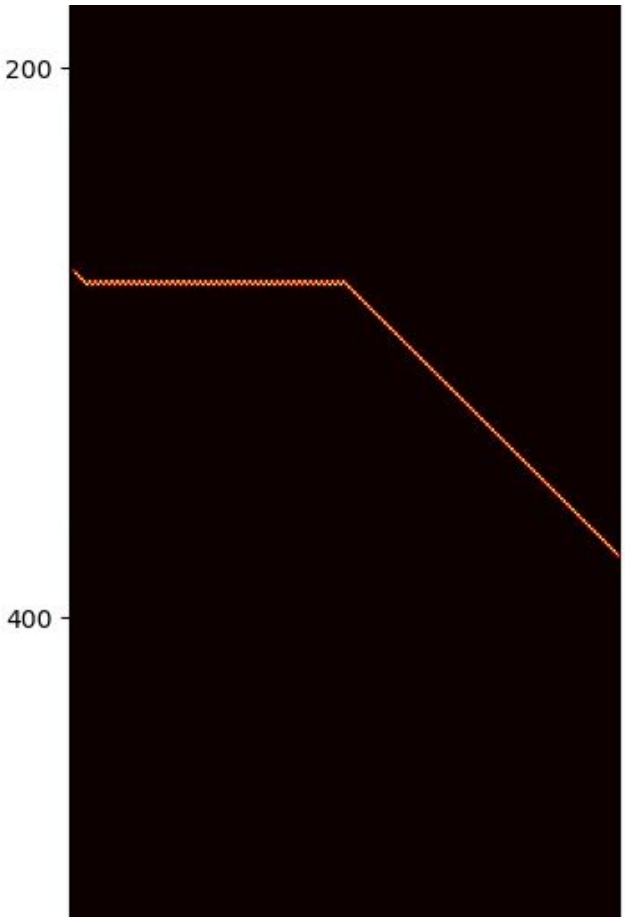


Figure 49: NTM memory read access for a correctly generalized instance of A_nB_n grammar. X axis is the sequence index and Y axis is the memory address.

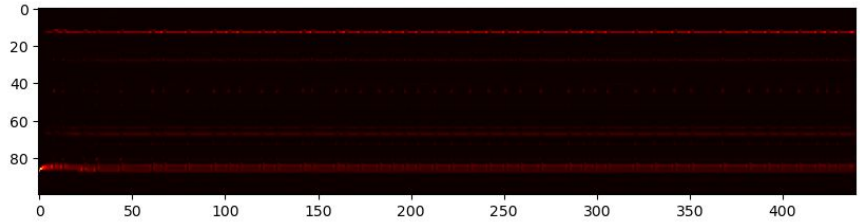


Figure 50: NTM memory read access for an incorrectly generalized instance of Reber grammar. X axis is the sequence index and Y axis is the memory address.

4.8 Investigation Based on Parameters

This section analyzes the impact of various training parameters on different models across diverse tasks.

4.8.1 Sequence Length

Figures 51, 52, and 53 illustrate the influence of maximum sequence length on the generalization rate of the models. Generally, an increase in sequence length positively affects the generalization rate.

Generalization rate is the ratio of generalization success under a certain circumstance. It can also be interpreted as the probability of generalization, other parameters being random. It can be defined as:

$$\text{Generalization Rate (parameter)} = \frac{\text{\# of generalized instances with parameter}}{\text{t\# of total instances with parameter}}$$

As depicted in Figure 51, for Reber Grammar and LSTM instances, successful generalization occurs regardless of the maximum sequence length. This implies that LSTMs are capable of generalizing the rules of a state machine, specifically the Reber Grammar, even with relatively short sequence lengths.

For A_nB_n Grammar, the generalization rate increases linearly with the maximum sequence length, suggesting a direct relationship between the two for LSTMs (up to a sequence length of 100). Sequences shorter than 50 failed to generalize, possibly due to the models encountering a local minimum in simpler examples.

LSTM's inability to generalize the Mirror Grammar indicates that the sequence length does not influence its Generalization Rate in this context.

For LMU samples, since neither A_nB_n nor Mirror Grammar was generalized, the maximum sequence length has no effect.

However, as seen in Figure 52, there is a strong positive correlation between maximum sequence length and generalization rate for Reber Grammar. It is noteworthy that there is no significant difference in generalization rate between sequence lengths of 70 and 100. This suggests that increasing the sequence

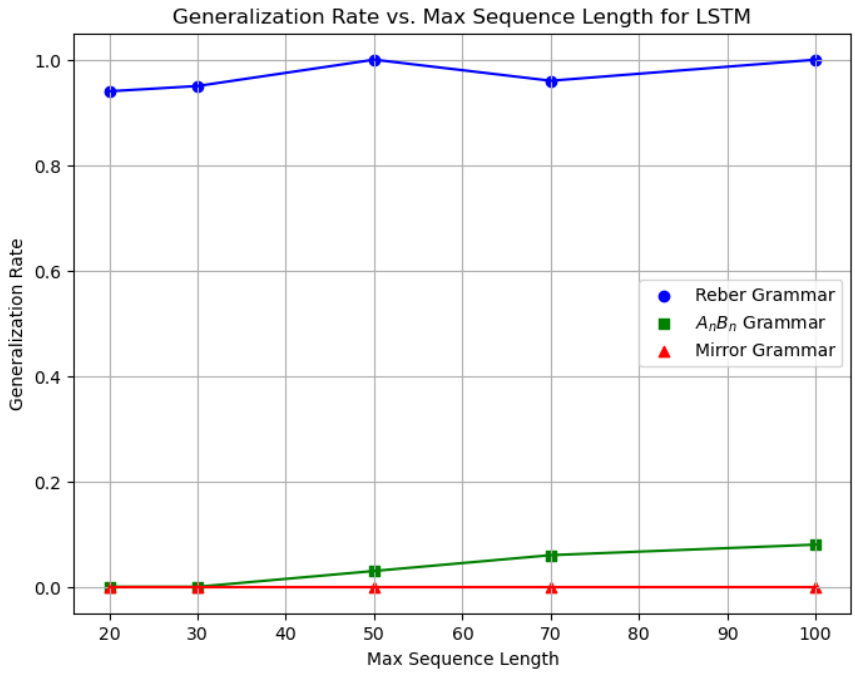


Figure 51: The effect of Maximum Sequence Length length on generalization rate for LSTM models

length up to 70 may improve LMUs' performance in learning Reber Grammar, but further increases do not yield additional benefits.

Turning to NTMs, Figure 53 reveals that the maximum sequence length has a minimal impact on generalization performance for both A_nB_n and Mirror Grammar tasks. There is only a slight but insignificant increase around 50 sequence length. This suggests that the generalization capability of NTMs in these tasks may be influenced more by other factors or even chance.

But for Reber Grammar, examples generalized better as the maximum length of the sequence increases.

4.8.2 Learning Rate

Figures 54, 55, and 56 indicate that learning rate either did not have a significant effect or yielded best results around 10^{-3}

4.8.3 Memory Specific Parameters

In this section, we examine parameters unique to certain models and their impact on Generalization Rate.

Figure 57 illustrates the influence of the Hidden Size parameter on the LMU model. It's evident that an increase in Hidden Size positively affects the LMU's performance.

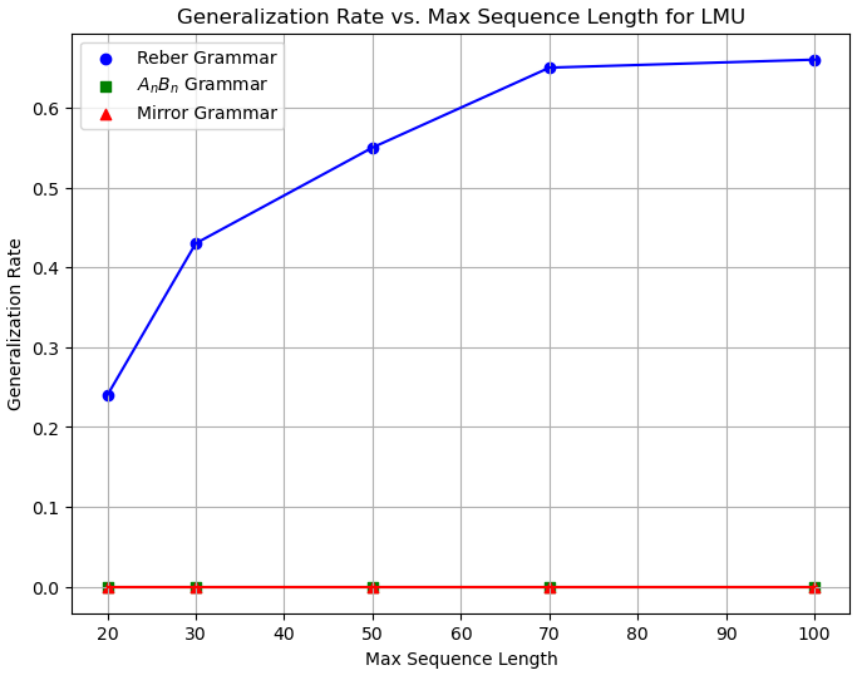


Figure 52: The effect of Maximum Sequence Length length on generalization rate for LMU models

Conversely, Figure 58 indicates that a larger LMU memory size negatively impacts the generalization rate. Similarly, the theta parameter of the LMU, as shown in Figure 59, also seems to have a detrimental effect on generalization performance.

These observations are understandable considering the nature of the experiment. The tokens used are simple and limited in number, suggesting that larger memories do not necessarily provide additional benefits. In fact, smaller memory sizes might be more advantageous, steering the training process towards rule learning rather than rote memorization. This aligns with the concept that optimal memory utilization, rather than sheer capacity, is crucial in effectively training neural network models.

Figure 60 shows that LSTM hidden size did not play a significant part in the performance

Moving to NTMs, Figure 62 suggests that an increase in memory width might have a detrimental effect on performance, similar to the LMUs. This could be due to the nature of the experiment, where larger memory may not necessarily aid in generalization and instead, could complicate the learning process.

Moreover, Figure 61 also depicts a complex relationship between the number of memory locations and generalization performance. For Reber Grammar, a very low number of memory locations had a higher generalization rate. Then as the memory location increases the performance dropped, whereas the memory locations increase, the generalization rate increases again. This might suggest there might be 2 different strategies involved in learning Reber grammar. One of them is more dependent on memory. But when the model is deprived of the memory it is forced to take the low memory strategy.

However, with the A_nB_n Grammar, there's an increased chance of generalization up to 1000 memory locations in our examples, but the ratio then declines. This decrease could be attributed to an unneces-

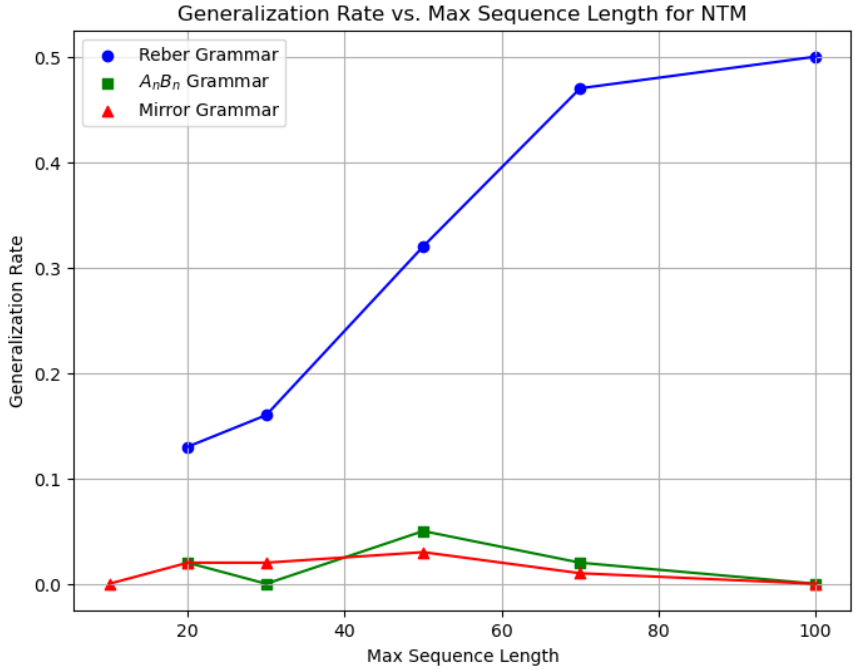


Figure 53: The effect of Maximum Sequence Length length on generalization rate for NTM models

sarily high number of memory locations, which might make the learning process more complex, or it could be a result of chance.

4.9 Investigation Based on Curriculum Learning

The results shown in Table 7 shows a combination of different grammars, model types and curriculum strategies. Table displays number of generalized instances (# of Gen) and Generalization rate (Gen. rate). Generalization rate is considered as the success criteria. Bold values show the generalization rate for the best curriculum for that grammar and model type combination.

Table 7: The experiments distribution among different curriculum learning strategies and how well they perform based on number of generalized instances (# of Gen) and Generalization rate (Gen. rate)

Grammar	Model Type	Curriculum	Cost Min	Cost Max	Cost Mean	Cost Count	# Gen.	Gen. rate
A_nB_n	LMU	Gradual Increase	2.25	759.75	456.62	21	0	0.0
		Grad. Inc. Spiking	3.00	872.00	465.14	74	0	0.0
		No Curriculum	380.00	765.00	643.22	22	0	0.0
		Uniform Random	111.00	893.00	452.16	83	0	0.0
	LSTM	Gradual Increase	3.00	870.00	378.79	14	0	0.0
		Grad. Inc. Spiking	0.00	1117.25	366.03	55	1	0.02
		No Curriculum	3.00	925.00	369.98	16	0	0.0
		Uniform Random	0.00	532.75	114.15	38	3	**0.08**
	NTM	Gradual Increase	2.00	817.50	289.57	55	0	0.0
		Grad. Inc. Spiking	1.00	746.50	266.30	106	0	0.0
		No Curriculum	2.00	894.00	271.13	60	0	0.0
		Uniform Random	0.00	932.75	238.88	143	9	**0.06**
Mirror	LMU	Gradual Increase	168.75	729.75	563.51	25	0	0.0
		Grad. Inc. Spiking	118.50	711.75	532.90	77	0	0.0
		No Curriculum	165.00	726.25	571.52	23	0	0.0
		Uniform Random	101.75	746.50	476.63	65	0	0.0
	LSTM	Gradual Increase	140.00	740.50	527.01	19	0	0.0
		Grad. Inc. Spiking	88.50	770.25	454.95	55	0	0.0
		No Curriculum	114.75	760.50	439.75	16	0	0.0
		Uniform Random	93.50	762.00	246.96	62	0	0.0
	NTM	Gradual Increase	105.25	886.50	602.48	487	0	0.0
		Grad. Inc. Spiking	0.00	768.25	458.69	242	3	**0.01**
		No Curriculum	134.25	774.50	426.71	81	0	0.0
		Uniform Random	0.00	750.00	313.78	224	2	**0.01**
Reber	LMU	Gradual Increase	0.00	331.20	59.38	19	9	0.47
		Grad. Inc. Spiking	0.00	234.80	51.26	54	27	0.5
		No Curriculum	0.00	232.20	39.76	17	5	0.29
		Uniform Random	0.00	188.20	26.04	65	36	**0.55**
	LSTM	Gradual Increase	0.00	112.80	7.52	15	14	0.93
		Grad. Inc. Spiking	0.00	23.80	1.52	31	29	0.94
		No Curriculum	0.00	0.00	0.00	7	7	**1.0**
		Uniform Random	0.00	0.00	0.00	57	57	**1.0**
	NTM	Gradual Increase	0.00	82.80	36.67	8	2	0.25
		Grad. Inc. Spiking	0.00	161.80	27.80	43	13	0.3
		No Curriculum	0.00	228.00	42.87	11	4	**0.36**
		Uniform Random	0.00	201.40	46.03	38	12	0.32

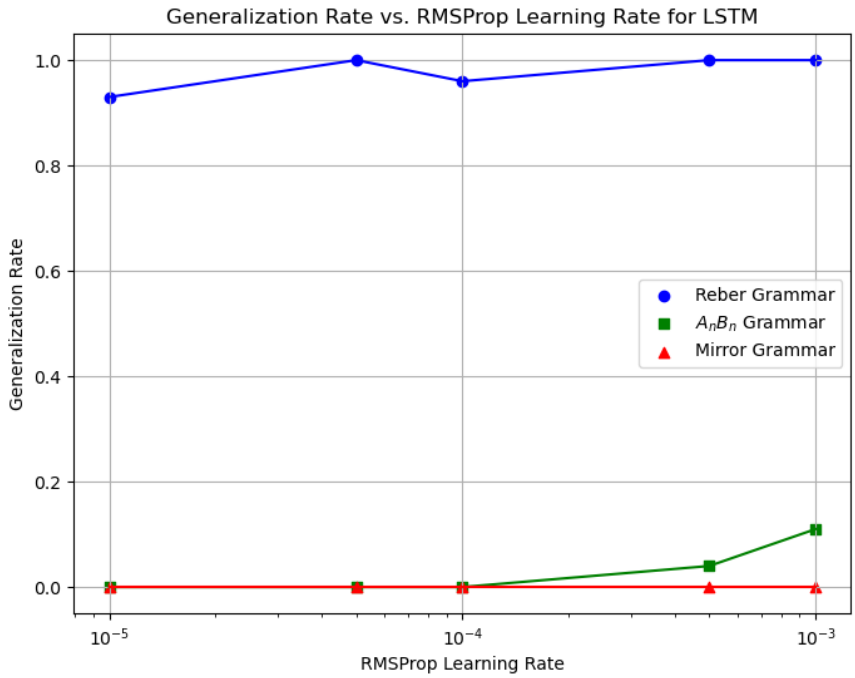


Figure 54: The effect of RMSProp Learning Rate length on generalization rate for LSTM models

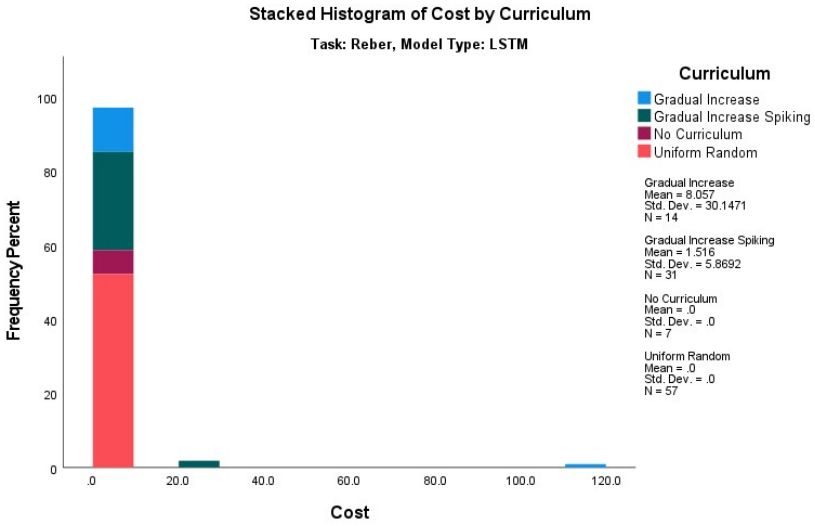


Figure 63: Stacked histogram displaying cost distribution of various curriculum type parameters for reber grammar and LSTM model

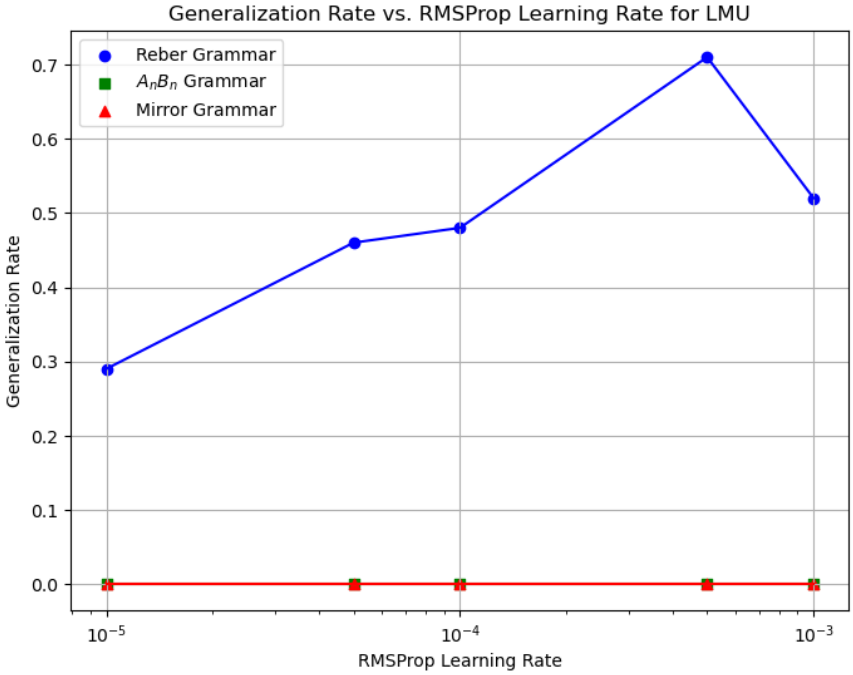


Figure 55: The effect of RMSProp Learning Rate length on generalization rate for LMU models

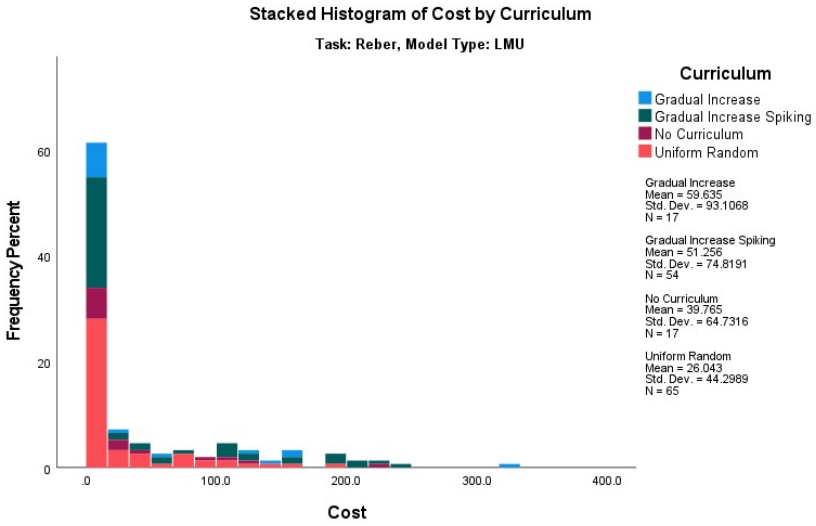


Figure 64: Stacked histogram displaying cost distribution of various curriculum type parameters for reber grammar and LMU model

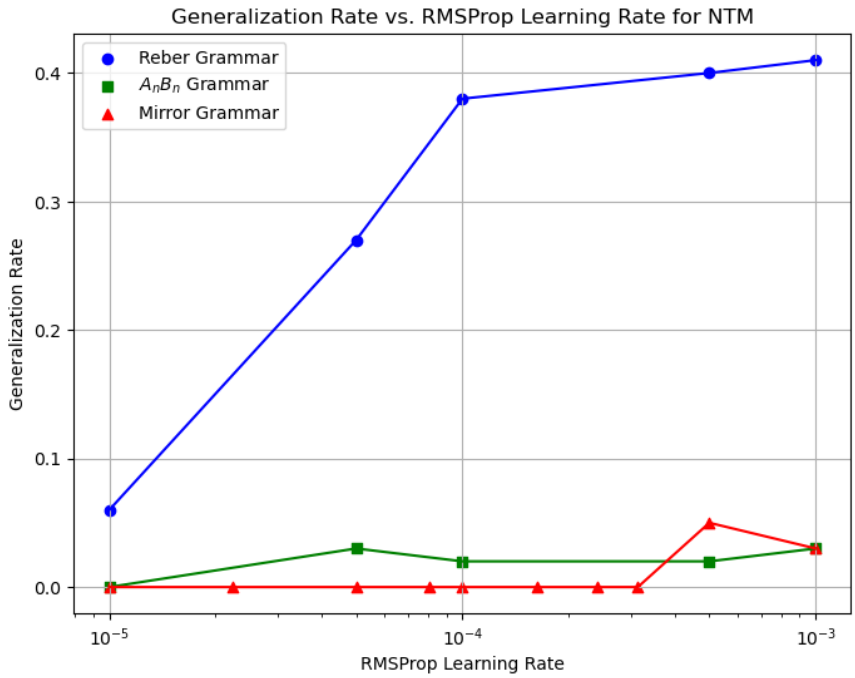


Figure 56: The effect of RMSProp Learning Rate length on generalization rate for NTM models

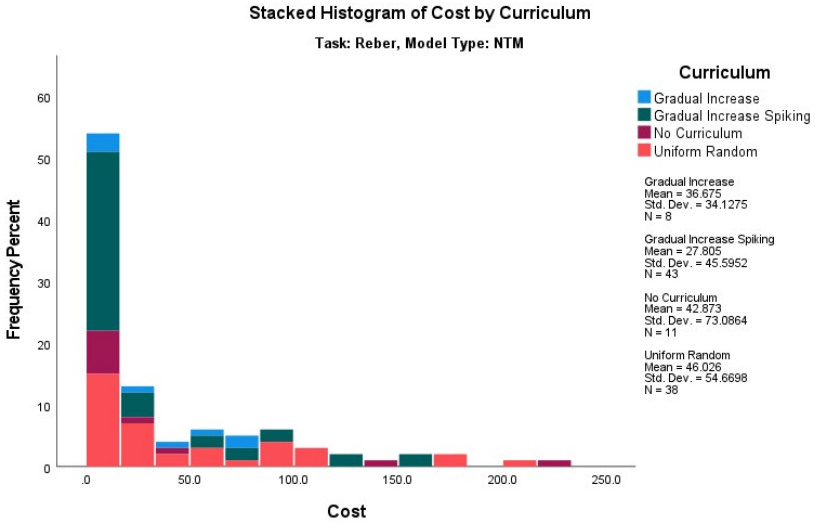


Figure 65: Stacked histogram displaying cost distribution of various curriculum type parameters for reber grammar and NTM model

Figures 63, 64, 65 show the stacked histogram for the cost values of the respective models trained on the Reber grammar for different curriculum types. The first bin close to 0 represents generalized instances or instances that had a very small cost.

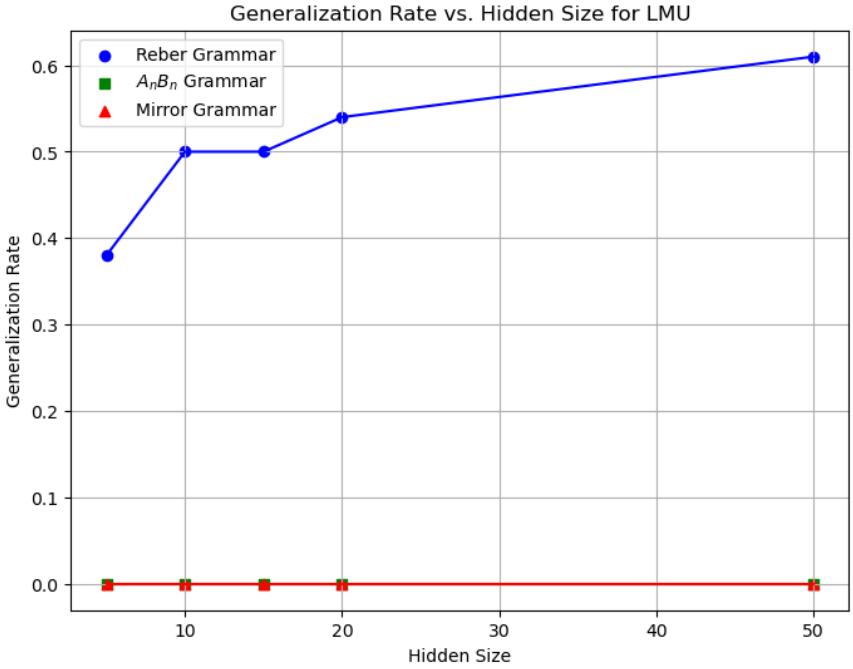


Figure 57: The effect of Hidden Size length on generalization rate for LMU models

LSTM can learn the sequences almost independent from the curriculum type indicating that curriculum type had little to no effect. For LMU no significance is observed except for No Curriculum that performs worst as seen in Table 7. For NTM there is not much observable difference.

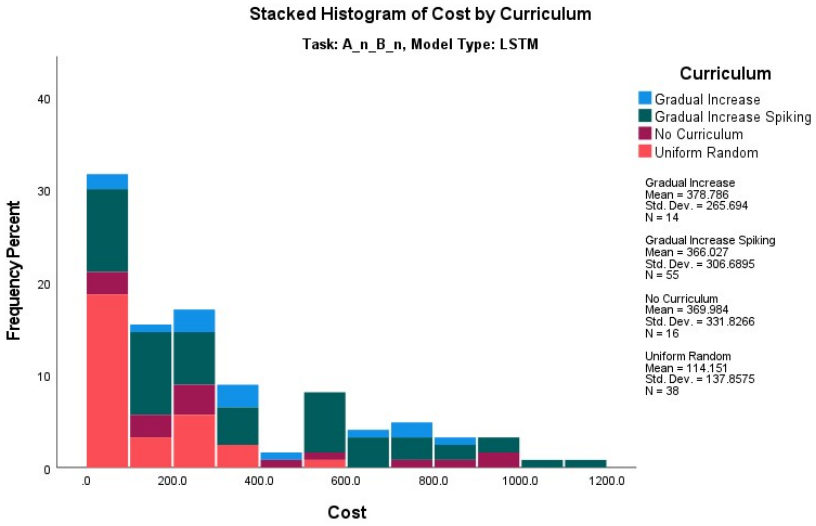


Figure 66: Stacked histogram displaying cost distribution of various curriculum type parameters for anbn grammar and LSTM model

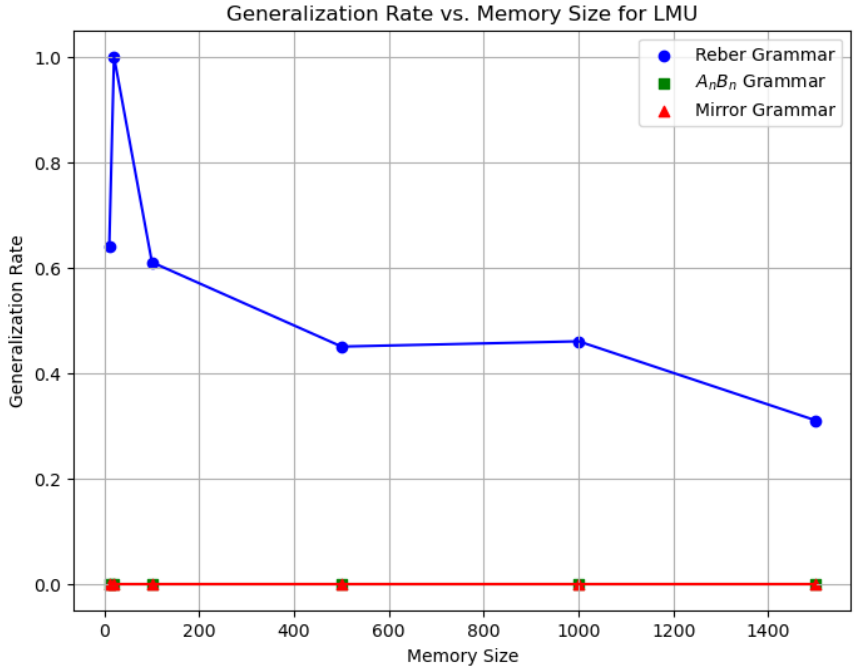


Figure 58: The effect of Memory Size length on generalization rate for LMU models

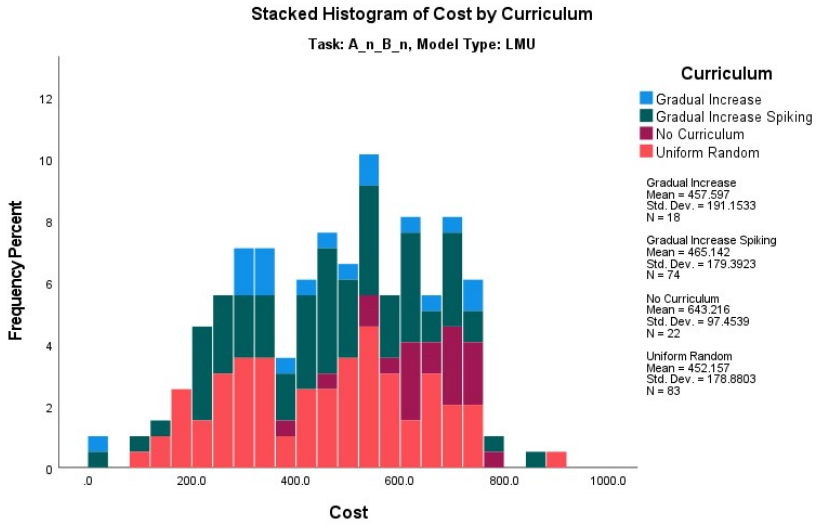


Figure 67: Stacked histogram displaying cost distribution of various curriculum type parameters for anbn grammar and LMU model

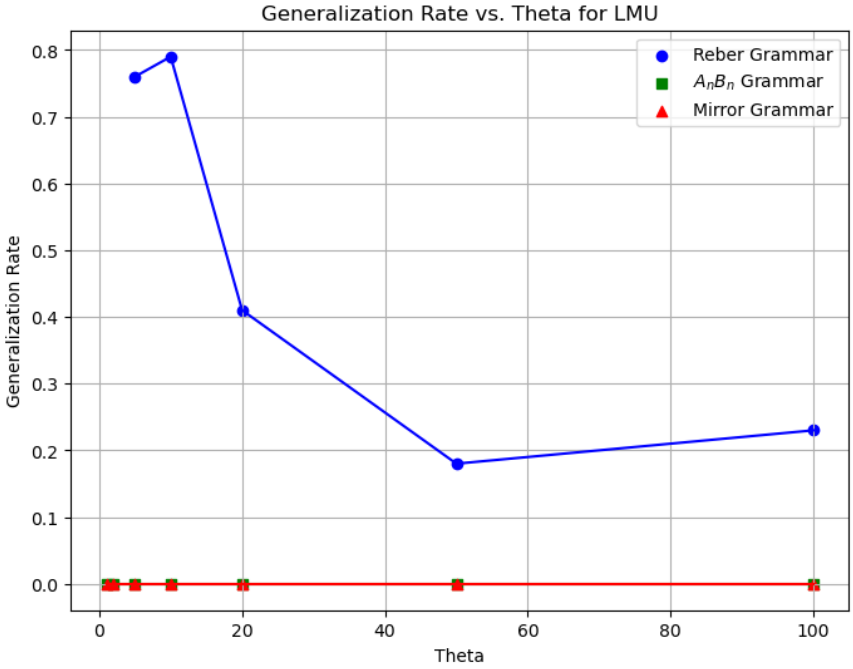


Figure 59: The effect of Theta length on generalization rate for LMU models

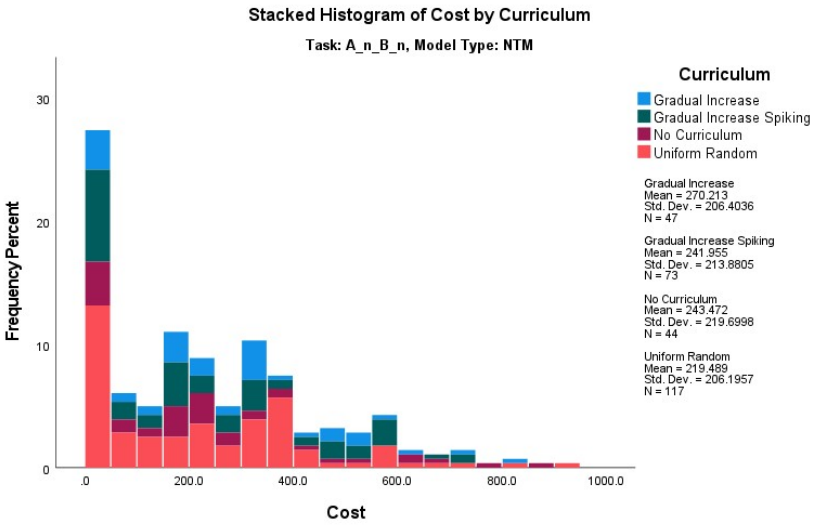


Figure 68: Stacked histogram displaying cost distribution of various curriculum type parameters for anbn grammar and NTM model

Figures 66, 67, 68 Show the Stacked histogram for cost values of respective models trained on A_nB_n grammar for different curriculum types. For the LSTM curriculum seems to have some effect where Uniform Random curriculum produced most of the low cost results. and as seen in Table 7, it has the best rate for generalization, however, low. No LMU model was able to generalize A_nB_n but some

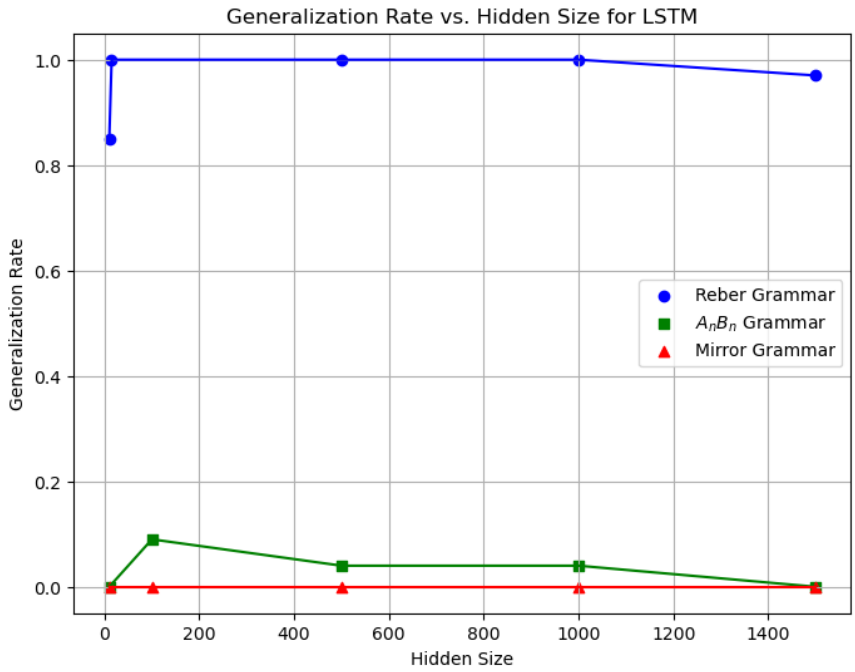


Figure 60: The effect of Hidden Size length on generalization rate for LSTM models

instances with Gradual Increase curriculum and Gradual Increase Spiking curriculum came close, but it might not have any significance. For NTM most of the instances that had a low cost were with Uniform Random curriculum. Similarly only instances with Uniform Random curriculum. could generalize.

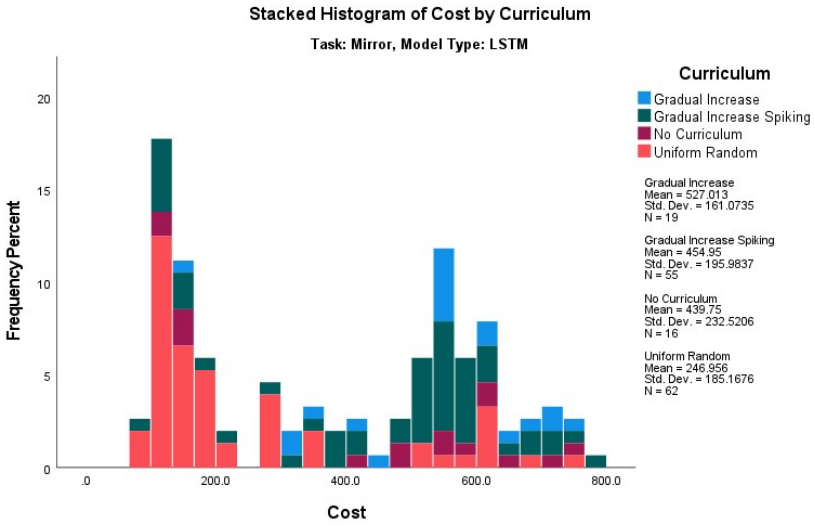


Figure 69: Stacked histogram displaying cost distribution of various curriculum type parameters for mirror grammar and LSTM model

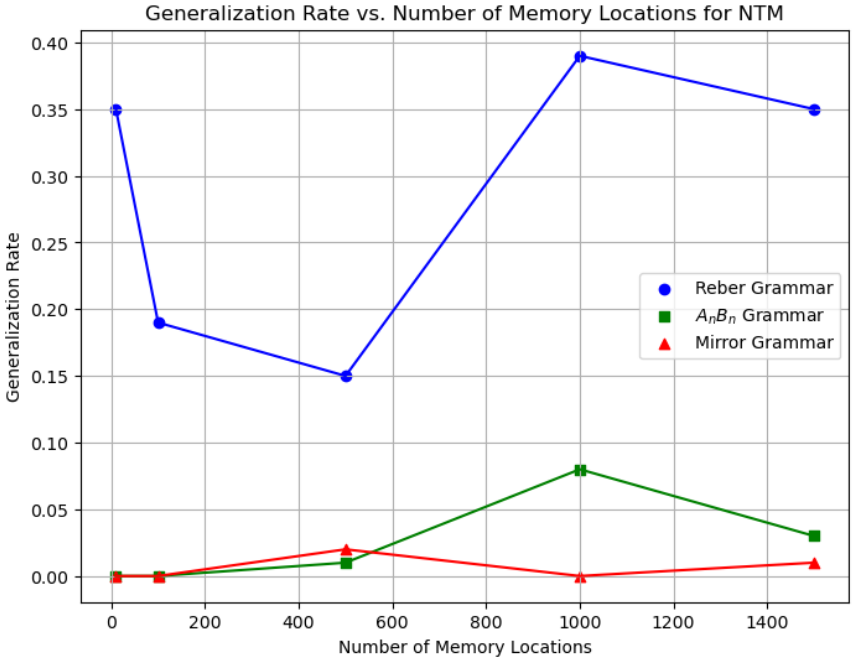


Figure 61: The effect of Number of Memory Locations length on generalization rate for NTM models

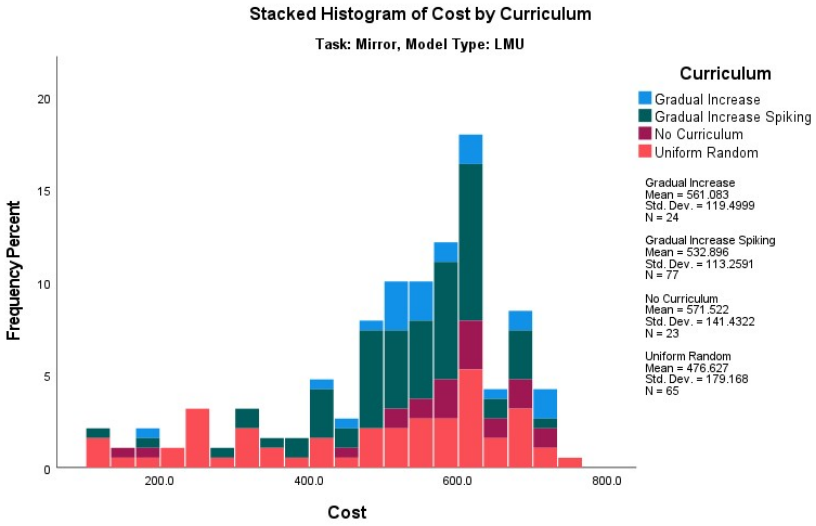


Figure 70: Stacked histogram displaying cost distribution of various curriculum type parameters for mirror grammar and LMU model

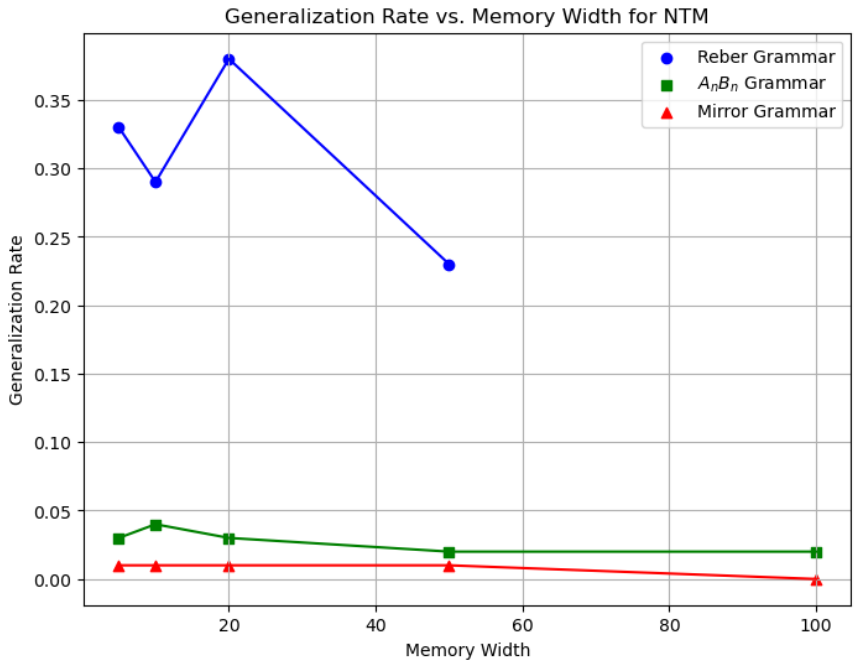


Figure 62: The effect of Memory Item width length on generalization rate for NTM models

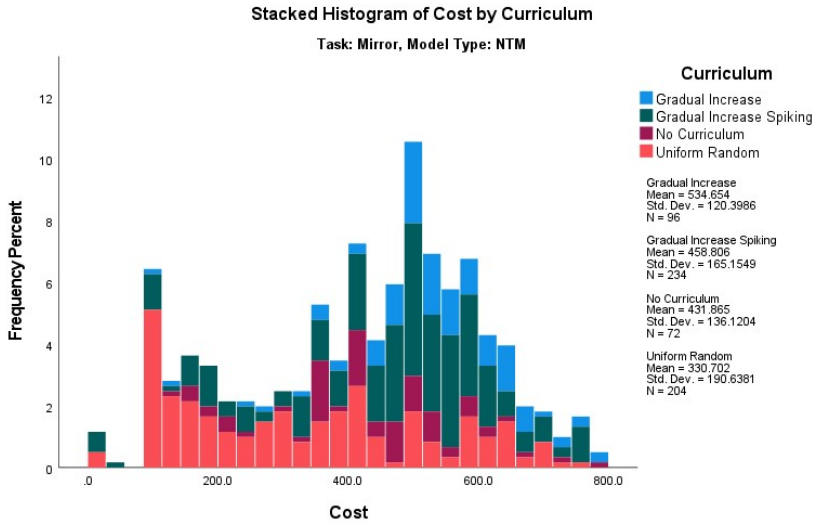


Figure 71: Stacked histogram displaying cost distribution of various curriculum type parameters for mirror grammar and NTM model

Figures 69, 70, 71 Show the results for mirror grammar. For LSTM no instance could generalize bur Uniform Random curriculum overall produced lower costs. LMU performed badly regardless of the Curriculum lowest among them being Uniform Random curriculum. For NTM only generalized instances were using Uniform Random or Gradual Increase Spiking curriculum. Generalization rates being low but similar as seen in Table 7.

Results indicate that Uniform Random Curriculum performs best overall with the exception of No Curriculum being better on generalization rate for Reber grammar and NTM. But this case might not be statistically significant since all other parameters are randomly chosen.

CHAPTER 5

MILDLY CONTEXT SENSITIVE GRAMMAR RESULTS

In this chapter, we will present the results obtained from mildly context sensitive grammar experiments. This chapter builds on the results of the previous chapter. In this chapter, we will adopt a more selective approach to look at generalization capabilities of MCSGs.

To evaluate the generalization capability we also tried a pre-training approach where a model was first trained in mirror (WW^R) grammar and the successful models were trained on the AWA (WW^R_WW^R) grammar which provided an easier way for generalization.

5.1 Evaluation Set

The trained models are evaluated on previously recorded datasets consisting of sequences of varying lengths. Similarly to previous experiments, like explained in Section 4.2, the main idea behind training sets is that they should contain sequences much larger than the original sequence length on which the models were trained on, that was limited to 100. The pre-train dataset evaluation does not hold importance in and on itself but is used for selection of the successful models that will be used for fine-tuning. Properties of these datasets can be seen in Table 8.

Then the cost for the dataset will be calculated on those evaluation datasets.

Table 8: The statistics of the evaluation set for various grammar types for MCSG experiments

Grammar Type	Min Seq. Len.	Max Seq. Len.	Mean Seq. Len.	# of Sequences
AWA Pre-train (WW^R)	50	500	212.5	4
AWA (WW^R_WW^R)	49	297	160.0	4
AWA-Star	57	353	192.5	4

5.2 Mildly Context Sensitive AWA Grammar Results

Additional tests were conducted using the AWA grammar, a type of Mildly Context-Sensitive Grammar (MCSG) described in Section 3.3.6.2. This grammar was used to test how well the models could generalize beyond context-sensitive grammars.

5.2.1 LSTM and LMU Performance

Neither the Long Short-Term Memory (LSTM) nor the Legendre Memory Unit (LMU) architectures were able to generalize the AWA Grammar task.

The AWA grammar, as explained in Section 3.3.6.2, combines a mirror grammar output with its replication. To learn this grammar, a network must be able to:

1. Generalize the mirror grammar (defined in Section 3.3.5.1)
2. Replicate the mirror grammar's output

As shown in Sections 4.6 and 4.3, only the Neural Turing Machine (NTM) was able to learn the mirror grammar. Since LSTM and LMU couldn't learn the mirror grammar, which is a key part of the AWA (WW^R_WW^R) grammar, they logically cannot learn the more complex AWA (WW^R_WW^R) grammar.

Due to this limitation, detailed experiments on the AWA grammar for LSTM and LMU models were not conducted. Instead, further tests focused on the NTM architecture, which had shown the necessary abilities to potentially generalize the AWA grammar.

5.2.2 NTM Performance

5.2.2.1 Pre-training Task (WW^R) Performance

The pre-training task was a very similar task to the mirror (WW^R) grammar; the only notable example is including another token 'W' before the grammar ends. For this reason performance of this grammar can already be seen from Section 4.6.

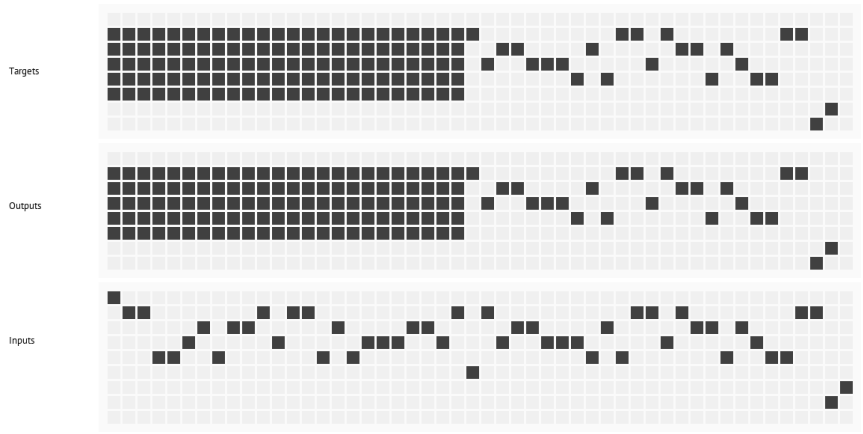


Figure 72: Input outputs and targets visualization for a generalized example for AWA Pre-train grammar

Generalized Instance Figure 72 shows the inputs outputs and targets for a generalized AWA Pre-train grammar. Figure 73 shows the NTM read head accesses for a generalized AWA Pre-train

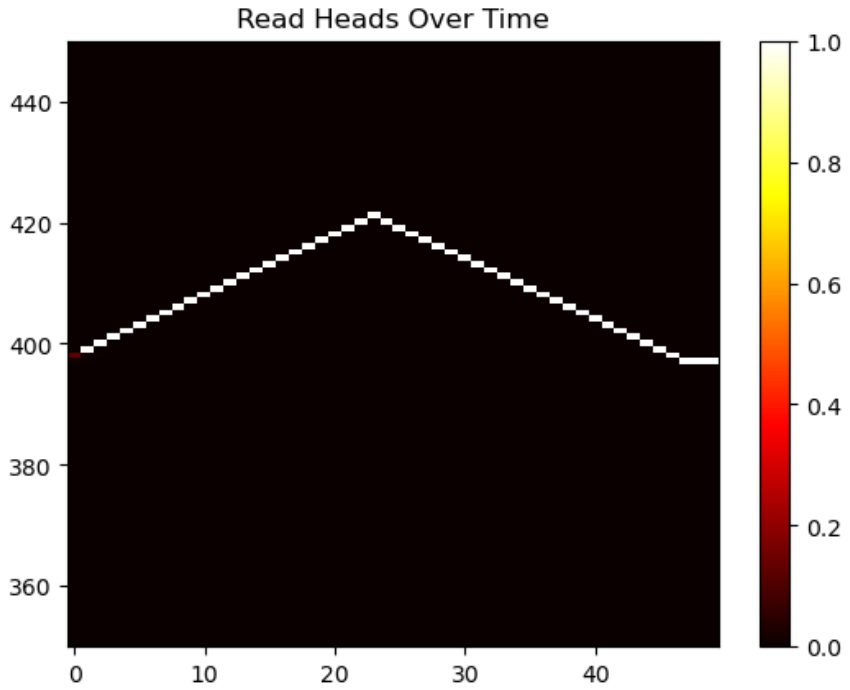


Figure 73: NTM Read Head visualization for a generalized example for AWA Pre-train grammar

grammar. Figure 74 shows the NTM write head accesses for a generalized AWA Pre-train grammar.

The model can be analysed based on Figures 72 73 74. First half of the pattern before the M character is written with A,B,C,D, and M characters possible because until M there is no specific rule for possible outputs except that it can't end without completing the mirror pattern. After the M character is given as input the model successfully predicts the reverse pattern. If we look at the read and write heads, write heads start to write at indices right below address 400 sequentially. When the M character is given as input, the read head successfully reads those items in reverse order.

This generalized model will be a starting point for fine-tuned AWA (WW^R_WW^R) grammars.

Non-generalized Instance Figure 75 shows the inputs outputs and targets for a non-generalized AWA Pre-train grammar. Figure 76 shows the NTM read head accesses for a non-generalized AWA Pre-train grammar. Figure 77 shows the NTM write head accesses for a non-generalized AWA Pre-train grammar.

The model can be analysed based on Figures 75 76 77. Similar to the generalized instance, the first half of the pattern before the M character is written with both A,B,C,D, and M characters possible because until M there is no specific rule for possible outputs except that it can't end without completing the mirror pattern. But after that the model does not provide any distinct possible output. It only learned which characters are possible before the M character. Read and write heads of the NTM show no sequential access pattern meaning that addressable memory was not utilized by this model.

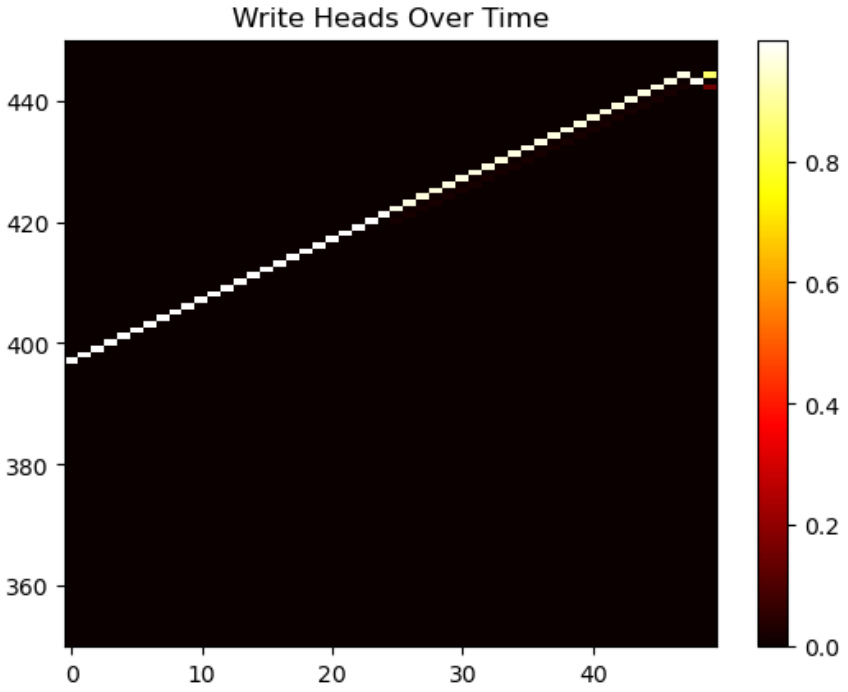


Figure 74: NTM Write Head visualization for a generalized example for AWA Pre-train grammar

5.2.2.2 AWA Grammar (WW^R_WW^R) Performance

This section examines the performance of the AWA (WW^R_WW^R) which is our main mildly context sensitive grammar. Most of the successful models are fine-tuned from the successful models explained in pre-training results explained in Section 5.2.2.1. Since we are using the same weight matrices in fine-tuning, parameters that affect the NTM controller, NTM memory are fixed. This fine-tuning approach does not allow us to select parameters freely. So this section will focus on properties of the generalized and non-generalized models rather than the parameters statistics.

These fine-tuning runs were made on up to a maximum sequence length of 100 and the trained models were tested on test sequences of up to 300 items. Generalization is determined by achieving zero cost on the test data set of out-of-distribution sequence examples. It should be noted that when the sequence length goes too extreme results may start to degrade because the NTM uses a limited amount of memory unlike the unlimited tape of Turing machines.

Generalized Instance Figure 78 shows the inputs outputs and targets for a generalized AWA (WW^R_WW^R) grammar. Figure 79 shows the NTM read head accesses for a generalized AWA (WW^R_WW^R) grammar. Figure 80 shows the NTM write head accesses for a generalized AWA (WW^R_WW^R) grammar.

The model can be analysed based on Figures 78 79 80. Model could successfully predict all the possible outputs of the grammar. In the middle of Figure 78 we can see the W character that starts the repetition part of the grammar. After that W character the inputs are repeated except for start and

Figure 75: Input outputs and targets visualization for a non-generalized example for AWA Pre-train grammar

end tokens. Successful generalization keeps all the properties of the pre-trained grammar but learns to repeat all the input tokens.

From the read and write access graphs in Figures 79 80, we can see that in the first 10 elements in the sequence the tokens are written to addresses around 500, then the downward trend indicates that the same addresses are read with reverse order, resulting in correct performance on reversing the substring. The model then chooses to write the reversed string again.

The strategy involved is always writing tokens in increasing order, then when the M or W character is inputted, it reads the written chunk in reverse order (W^R) and starts writing again. But since the sub-string (W as the word) is written to addresses around 500 second write operation is not performed by the model But still it can read from the same addresses.

Generalized Longer Instance Figure 81 shows the inputs outputs and targets for a generalized-long AWA ($WW^R\_WW^R$) grammar. Figure 82 shows the NTM read head accesses for a generalized-long AWA ($WW^R\_WW^R$) grammar. Figure 83 shows the NTM write head accesses for a generalized-long AWA ($WW^R\_WW^R$) grammar.

The model can be analysed based on Figures 81 82 83. This example is for a generalized model running on a longer sequence. The model was able to perform without a fault in the shown sequence. Memory graphs in Figures 82 83 indicate it applies a similar strategy to the previous generalized model shown in Figures 79 80. Only notable difference being after the start of the repetition, it starts writing tokens to a new memory location around 500 at around the 100th item in the sequence.

Non-Generalized Instance Figure 84 shows the inputs outputs and targets for a non-generalized AWA ($WW^R\_WW^R$) grammar. Figure 85 shows the NTM read head accesses for a non-generalized AWA ($WW^R\_WW^R$) grammar. Figure 86 shows the NTM write head accesses for a non-generalized AWA ($WW^R\_WW^R$) grammar.

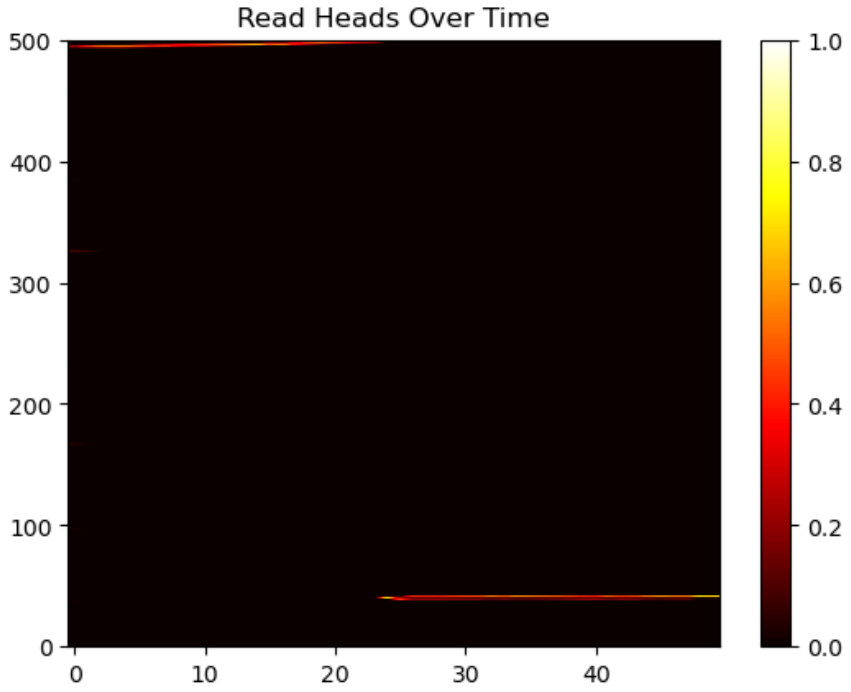


Figure 76: NTM Read Head visualization for a non-generalized example for AWA Pre-train grammar

The model can be analysed based on Figures 84 85 86. This example shows a non-generalized AWA (WW^R_WW^R) instance. The only rule it could learn was that all outputs are possible until the M character is given. After the M token is received to start mirroring, the read access switches to more content-based access than address-based, as indicated by the blurred read activations shown in Figure 85.

5.3 Mildly Context Sensitive AWA-Star WW^R_WW^R(_WW^R)* Grammar Results

In addition to tests performed on AWA (WW^R_WW^R) Grammar that is presented at Section 5.2, more tests were conducted using the AWA-Star (WW^R_WW^R(_WW^R)*) grammar, a type of Mildly Context-Sensitive Grammar (MCSG) described in Section 3.3.6.3. This grammar is an extension and slightly more complex version of the AWA (WW^R_WW^R) grammar.

The NTM was able to generalize and provide satisfactory results with this grammar similar to the AWA (WW^R_WW^R) grammar.

5.3.1 LSTM and LMU results

This grammar includes the mirror grammar (WW^R) as a part of the grammar. Similar to the AWA grammar explained in Section 5.2.1 LMU and LSTM models were not able to generalize this grammar due to their inability to learn the rules of mirror grammar.

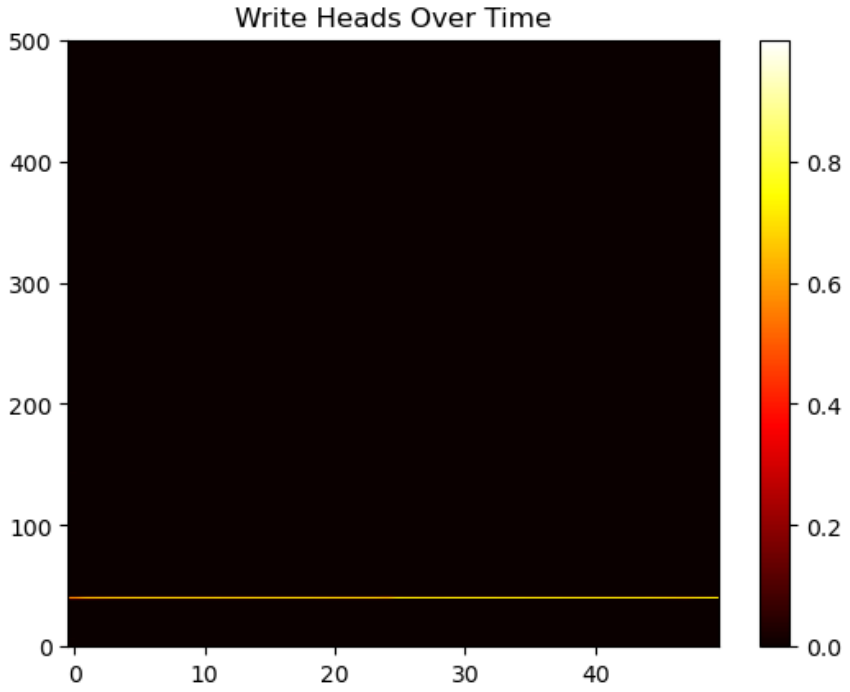


Figure 77: NTM Write Head visualization for a non-generalized example for AWA Pre-train grammar

5.3.2 NTM Results

Generalized Instance Figure 87 shows the inputs outputs and targets for a generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 88 shows the NTM read head accesses for a generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 89 shows the NTM write head accesses for a generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar.

The model can be analysed based on Figures 87 88 89. This example includes 5 repetitions of the WW^Rpattern. The results are very similar to the AWA (WW^R_WW^R) grammar results explained in Section 5.2.2.2. As we can see in Figure 87, the main difference that can be seen in the input and output patterns is that it has many more repetitions and a generalized model was able to handle it successfully. When we look at the model memory read patterns from Figures 88 and 89, we can see that the model generally writes the incoming tokens to memory in a sequential pattern but chooses to mostly read the first addresses around 840.

Generalized Longer Instance Figure 90 shows the inputs outputs and targets for a generalized-long AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 91 shows the NTM read head accesses for a generalized-long AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 92 shows the NTM write head accesses for a generalized-long AWA-Star (WW^R_WW^R(_WW^R)*) grammar.

The model can be analysed based on Figures 90 91 92. This example is similar to the previous generalized example, but includes a longer sequence as can be seen in Figure 90. Memory accesses shown in Figures 91 and 92 indicate a clearer read and write pattern where the model does make use of specific

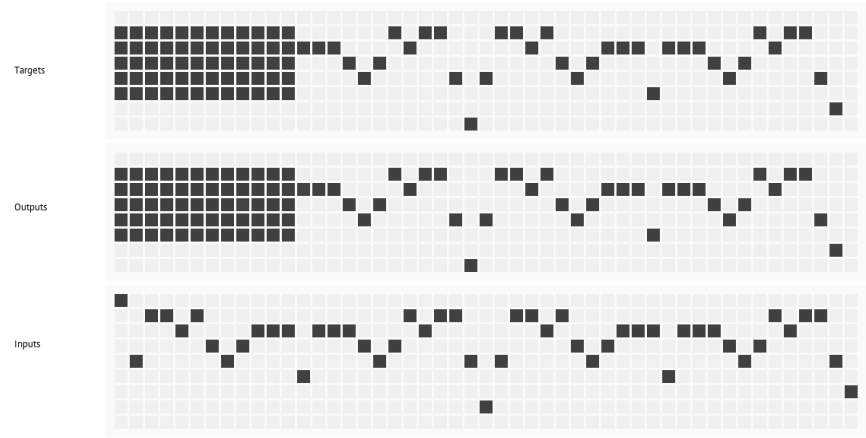


Figure 78: Input outputs and targets visualization for a generalized example for AWA (WW^R_WW^R) grammar

memory addresses until the M character is received at around the 10th token. From that point memory is read in an increasing order until a W character is inputted. Then the memory readings shift double the size of the repeated word. The following strategy can be derived from the memory accesses.

- Start reading from the last written address in the reverse order when first M (reverse token) is received
- reset the address offset to end of first repetition
- read from that address until W (repeat) token is received

Non-generalized Instance Figure 93 shows the inputs outputs and targets for a non-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 94 shows the NTM read head accesses for a non-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 95 shows the NTM write head accesses for a non-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar.

The model can be analysed based on Figures 93 94 95. This example could not generalize the grammar. But the behavior indicates that it still retained the learned rules from the mirror grammar that it was pre-trained on. Figure 94 indicates an address-based memory access until the end of first repetition, then the memory access becomes more content-based and the model begins to give wrong outputs.

Nearly-generalized Instance Figure 96 shows the inputs outputs and targets for a nearly-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 97 shows the NTM read head accesses for a nearly-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar. Figure 98 shows the NTM write head accesses for a nearly-generalized AWA-Star (WW^R_WW^R(_WW^R)*) grammar.

The model can be analysed based on Figures 96 97 98. This example is a model that could nearly generalize the rules with an exception. After the first M is given as input that starts the reversal of a given word, it fails to access the correct memory address for the first token after M. Rest of the grammar

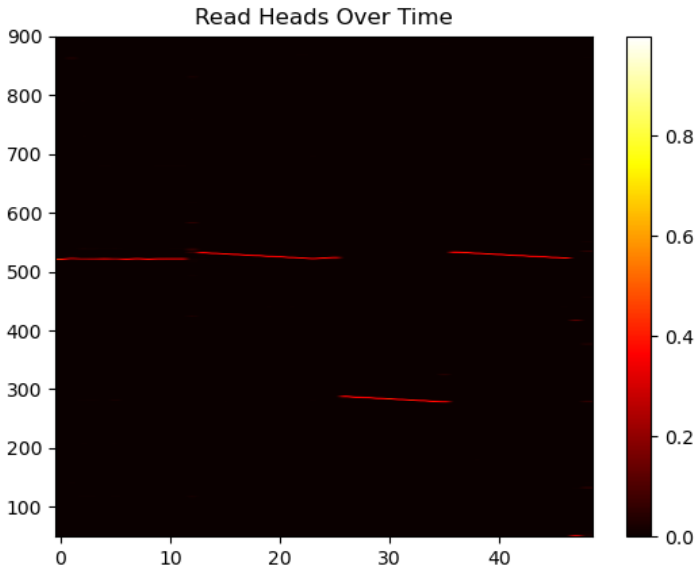


Figure 79: NTM Read Head visualization for a generalized example for AWA (WW^R_WW^R) grammar

runs as intended. This presents an example in which the adopted strategy may have drawbacks and fails to generalize completely.

5.4 Overall Evaluation of Mildly Context Sensitive Grammar Results

The simulation experiments performed for mildly context-sensitive grammars yields the following results.

- LSTM and LMU models could not generalize the AWA (WW^R_WW^R) grammar.
- NTM was able to generalize the AWA (WW^R_WW^R) grammar.
- LSTM and LMU models could not generalize the AWA-Star (WW^R_WW^R(_WW^R)*) grammar.
- NTM was able to generalize the AWA-Star (WW^R_WW^R(_WW^R)*) grammar.
- Pre-training the models on Mirror (WW^R) grammar led to better generalization of AWA (WW^R_WW^R) grammar and AWA-Star (WW^R_WW^R(_WW^R)*) grammar.
- Generalized models tend to have sharper memory access patterns that achieved by the location based addressing at NTMs
- It is possible to evaluate the algorithm performed by the NTM looking at memory patterns.

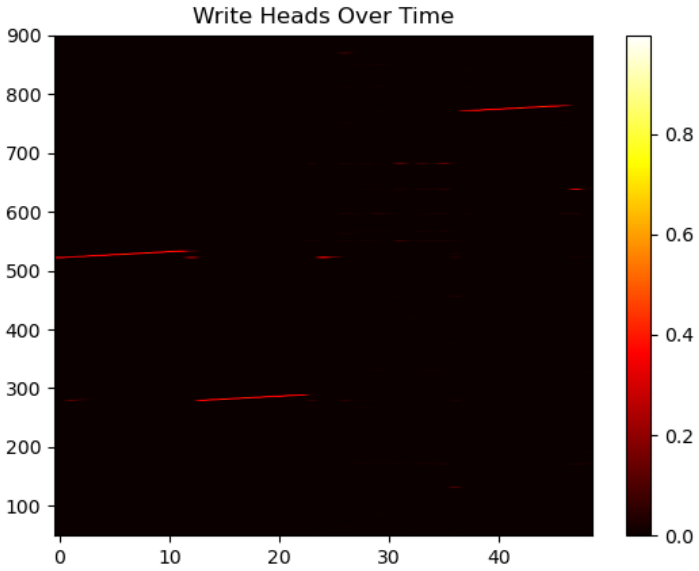


Figure 80: NTM Write Head visualization for a generalized example for AWA (WW^R_WW^R) grammar

Figure 81: Input outputs and targets visualization for a generalized model, long input example for AWA (WW^R_WW^R) grammar

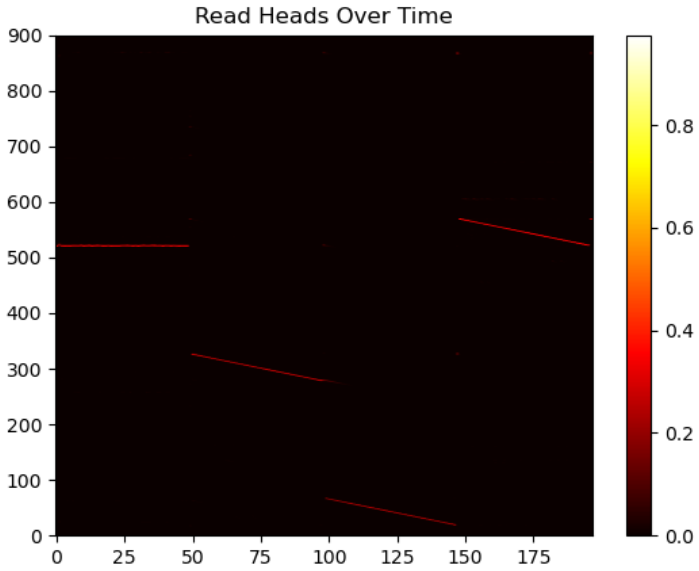


Figure 82: NTM Read Head visualization for a generalized model, long input example for AWA (WW^R_WW^R) grammar

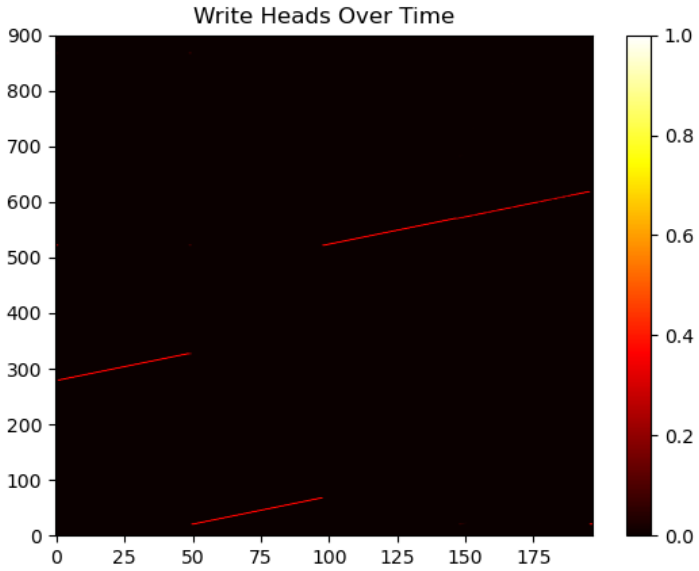


Figure 83: NTM Write Head visualization for a generalized model, long input example for AWA (WW^R_WW^R) grammar

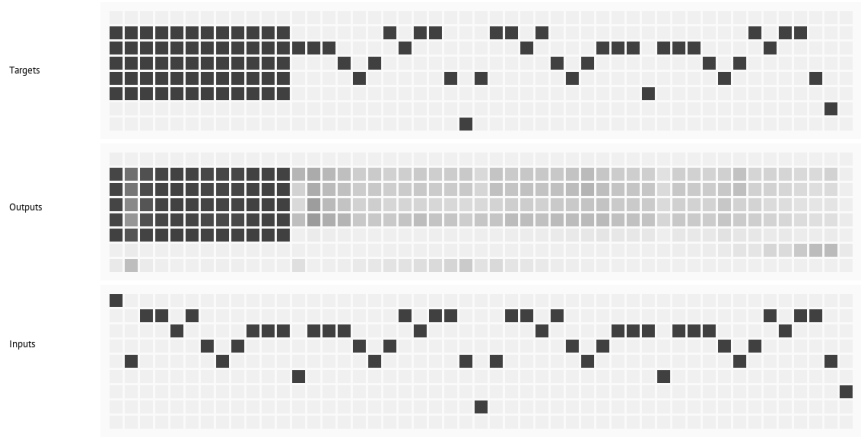


Figure 84: Input outputs and targets visualization for a non-generalized example for AWA (WW^R_WW^R) grammar

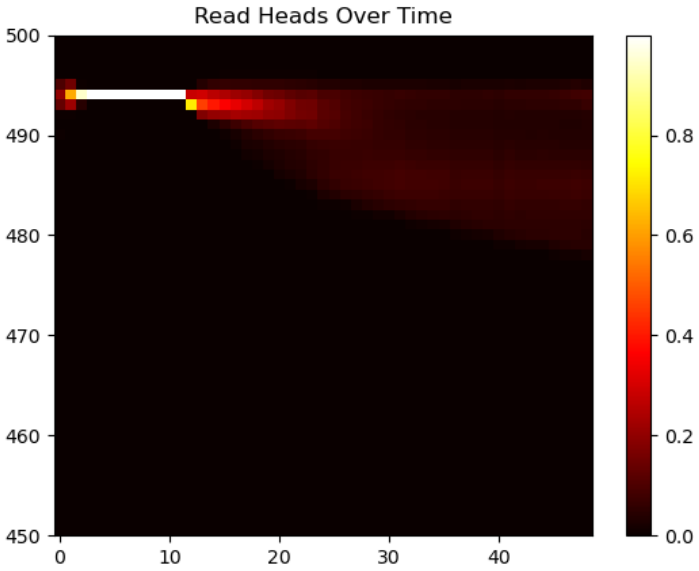


Figure 85: NTM Read Head visualization for a non-generalized example for AWA (WW^R_WW^R) grammar

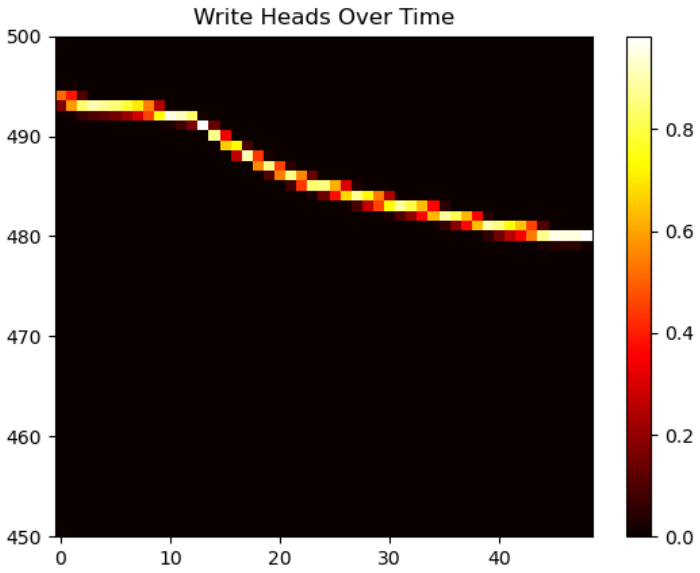


Figure 86: NTM Write Head visualization for a non-generalized example for AWA (WW^R_WW^R) grammar

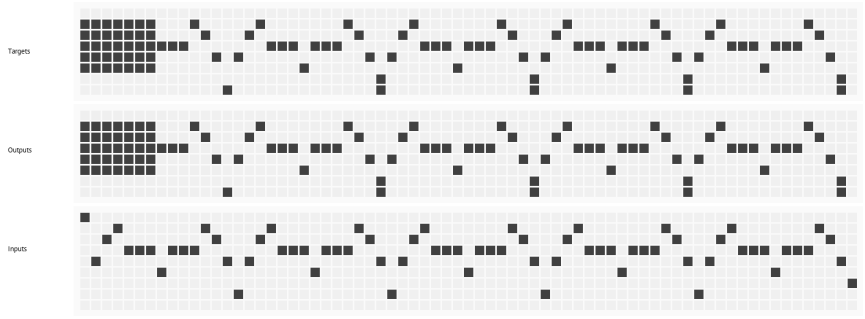


Figure 87: Input outputs and targets visualization for a generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

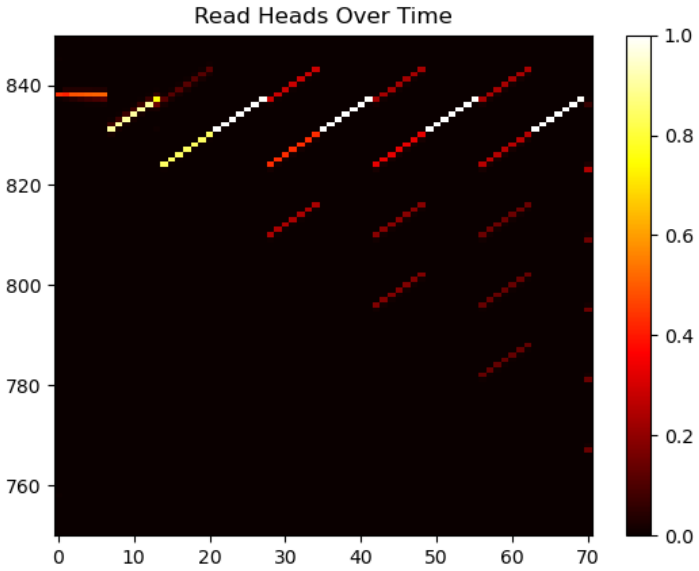


Figure 88: NTM Read Head visualization for a generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

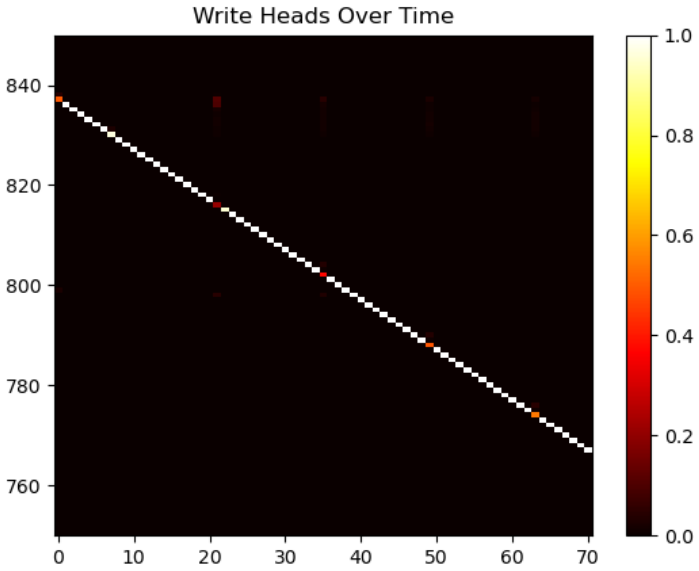


Figure 89: NTM Write Head visualization for a generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

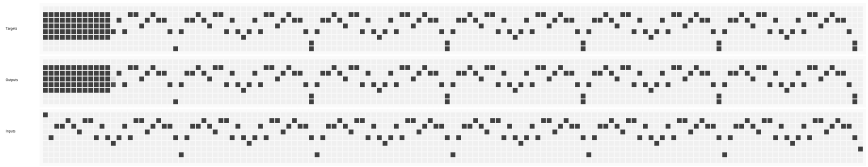


Figure 90: Input outputs and targets visualization for a generalized model, long input example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

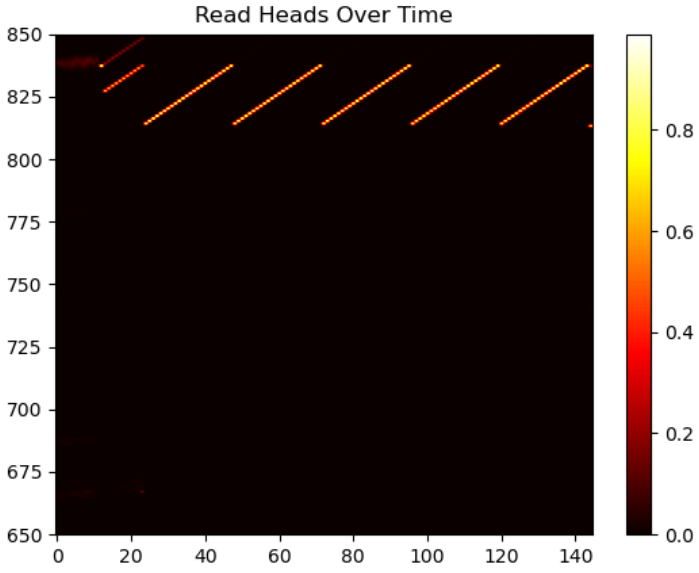


Figure 91: NTM Read Head visualization for a generalized model, long input example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

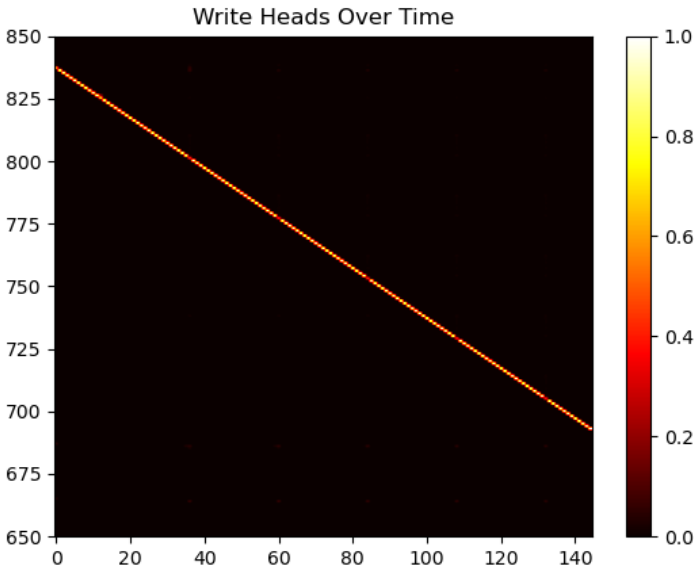


Figure 92: NTM Write Head visualization for a generalized model, long input example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

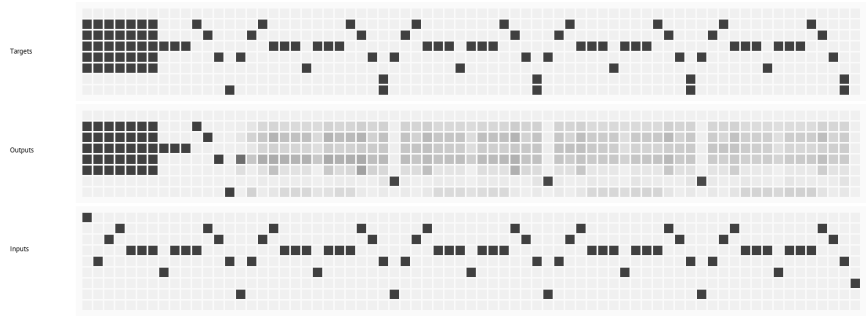


Figure 93: Input outputs and targets visualization for a non-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

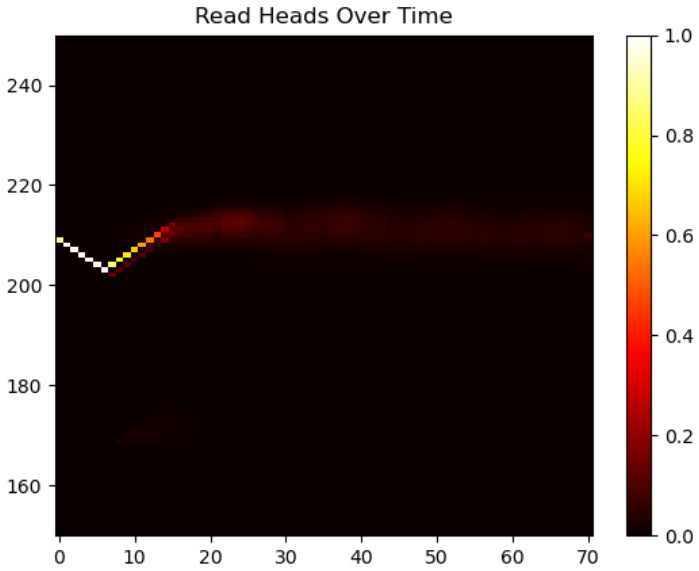


Figure 94: NTM Read Head visualization for a non-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

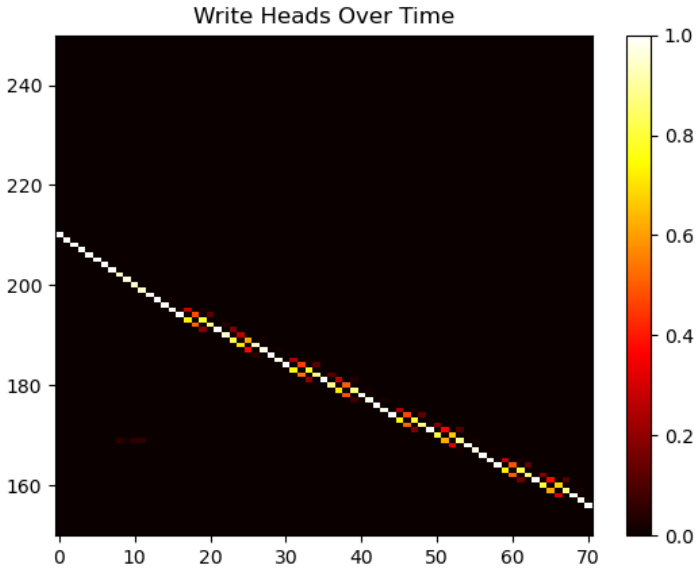


Figure 95: NTM Write Head visualization for a non-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

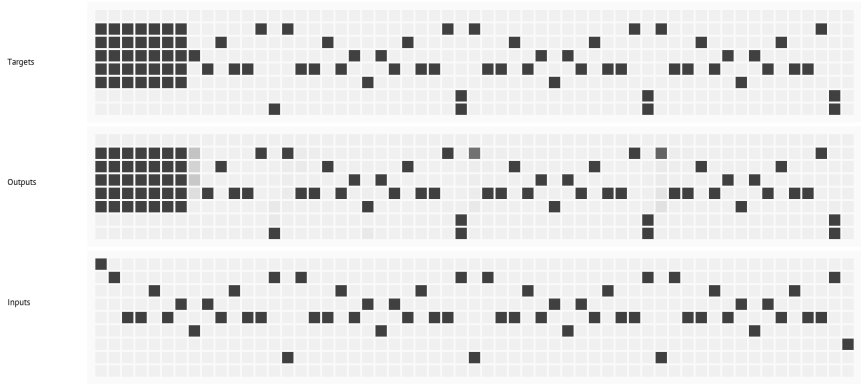


Figure 96: Input outputs and targets visualization for a nearly-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

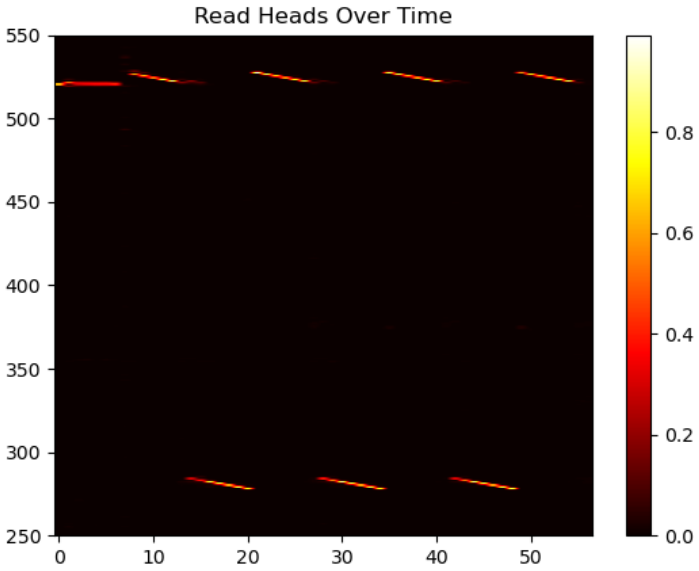


Figure 97: NTM Read Head visualization for a nearly-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

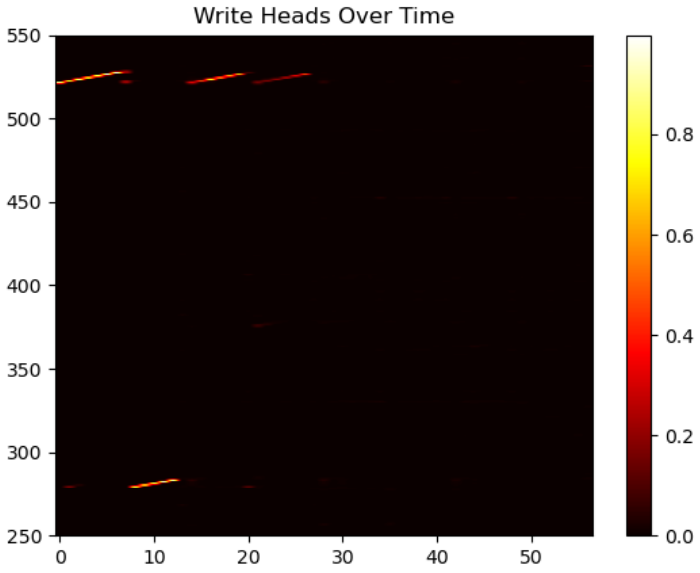


Figure 98: NTM Write Head visualization for a nearly-generalized example for AWA-Star (WW^R_WW^R(_WW^R)*) grammar

CHAPTER 6

DISCUSSION

This chapter integrates the findings from our extensive experiments on various artificial grammars with related studies in the field, providing a comprehensive analysis of memory networks as neuro-computational models of cognitive functions. Our research, centered on detailed investigations of rule learning capabilities and generalization performance across different grammatical complexities, forms the foundation of this discussion.

We designed and conducted a series of rigorous experiments to evaluate how different neural network architectures—specifically LSTM, LMU, and NTM models—learn and generalize rules from artificial grammars of varying complexity. These experiments were carefully crafted to probe the limits of each architecture's ability to internalize grammatical structures and apply them to novel sequences, providing insights into their potential as models of human syntactic processing.

While our study offers an in depth exploration into specific architectures and grammatical structures, the implications of our findings can be extended and enriched by considering them alongside other work in the field. Therefore, this chapter will also encompass a broader range of neural network architectures, including transformers, stack-augmented networks, and tape-based mechanisms. By examining how these diverse architectures handle tasks of varying grammatical complexity, we aim to construct a more comprehensive picture of the computational requirements for different levels of language processing in the brain.

The following sections will:

- Present a detailed analysis of our experimental results on artificial grammar learning, highlighting key findings on rule acquisition and generalization across different network architectures.
- Contextualize our results within the broader landscape of computational linguistics and cognitive modeling by examining related artificial grammar learning experiments.
- Explore the cognitive science implications of our findings and related work, focusing on how different neural architectures align with theories about human language.
- Investigate why memory networks, especially NTMs, demonstrate superior performance in generalizing mildly context-sensitive grammars, and discuss the implications for understanding human language capabilities.
- Draw connections between our computational models and neuroscientific understanding of language and memory systems in the brain.

- Evaluate how our computational modeling and simulation framework contributes to bridging the gap between connectionist and symbolic processing paradigms in cognitive science.

In this section, we aim not only to summarize our results and contributions but also to discuss the broader implications of these findings by integrating insights from related work in an effort to contribute to our understanding of the computational principles underlying syntactic processing and their neural correlates.

6.1 Related Artificial Grammar Work

This section will go over related research about learning artificial grammars with different neural network architectures and how they were able to learn and generalize various complexities of tasks and grammars.These studies include different neural network architectures. Some make direct use of LSTM like this thesis, or similar recurrent neural networks like GRU (Gated Recurrent Unit). These studies also propose different approaches for handling memory such as various addressable memory structures or integrated stack structures.

Even though these research papers were not written from a cognitive science perspective, further analyses about these tasks and their cognitive science implications are discussed in this section. Later in the chapter, these results will be combined with the results from this thesis to form a more holistic and detailed picture.

6.1.1 Recurrent Neural Networks with External Addressable Memory

The study by Quan et al. [39] introduced a novel architecture called External Addressable Long-Term and Working Memory (EALWM) for Recurrent Neural Networks. This architecture was designed to address the challenge of learning long-term dependencies in sequential data, a problem that is central to the current thesis as well.

EALWM separates short-term and long-term memory mechanisms, both of which are externally addressable. This approach is claimed to be inspired by human cognitive systems [39]. The separation of memory types in EALWM bears similarities to the distinct roles of working memory and episodic memory in human cognition, a concept that is explored in the present work through the lens of Neural Turing Machines (NTMs), Long Short-Term Memory networks (LSTMs), and Legendre Memory Units (LMUs).

Quan et al. demonstrated that EALWM can learn long-term dependencies without suffering from the vanishing gradient problem [39]. This capability is crucial for processing complex sequential data, including the artificial grammars studied in the current thesis. While EALWM was not specifically tested on artificial grammar learning tasks, its performance on algorithm learning tasks such as repeat copy, associative recall, and priority sort suggests potential applicability to artificial grammar processing.

The EALWM architecture's performance was compared to that of NTMs on these algorithm learning tasks. While EALWM was reported to outperform NTMs, it is important to note that the comparison may not have been entirely equitable. The NTM's memory size was kept equal to EALWM's working

memory size, while EALWM had additional memory for long-term storage [39]. This discrepancy in total memory capacity could have influenced the performance differences. In contrast, the current thesis aims to provide a more balanced comparison between NTMs, LSTMs, and LMUs by equalizing their *effective* memory capacities.

In terms of simillarity to cognitive systems, EALWM's separation of short-term and long-term memory aligns with cognitive theories of human memory systems. However, the current thesis goes further in exploring the biological plausibility of memory mechanisms by examining how different neural network architectures might correspond to specific brain regions and cognitive processes involved in language processing.

In addition to algorithm learning tasks, EALWM was evaluated on natural language processing tasks, including language modeling and question answering. The results indicated that external memory mechanisms enhance performance on tasks requiring long-term dependencies in natural language [39]. While the current thesis focuses on artificial grammar learning rather than natural language tasks, both studies share an interest in understanding how neural networks can process and learn complex sequential patterns.

One limitation of the EALWM study in relation to the current thesis is its focus on practical performance rather than on understanding the specific computational properties that enable grammar learning. The present work aims to fill this gap by analyzing how different memory mechanisms in neural networks contribute to the *learning* and *generalization* of grammatical rules.

The success of EALWM in handling long-term dependencies provides further evidence for the potential of external memory mechanisms in neural network architectures, a key aspect of the present research. Future work could explore how insights from EALWM might be integrated with the findings of the current thesis to develop more biologically plausible models of language processing and artificial grammar learning.

6.1.2 Neural State Pushdown Automaton

The work by Mali et al. [40] on Neural State Pushdown Automata (NSPDA) shows several important parallels to the research conducted in this thesis. Both studies explore the capabilities of neural networks augmented with external memory structures to learn and recognize complex grammatical patterns.

The NSPDA combines a digital stack with a neural network state machine, which is conceptually similar to the memory-augmented neural architectures examined in this thesis. A key finding from the study is that RNNs coupled with external memory, specifically a discrete stack, were able to generalize beyond the length of training examples when recognizing context-free grammars (CFGs) such as palindrome, $a^n b^n$, $a^n b^n c b^m a^m$, $a^{n+m} b^n c^m$, parenthesis grammars. This aligns closely with our observations on the generalization capabilities of memory-augmented neural networks.

Mali et al. [40] reported that LSTM and GRU models could perform well on some CFG tasks due to their ability to perform dynamic counting. However, these models struggled to learn proper state representations and had difficulty generalizing to longer sequences. This is consistent with our findings regarding the limitations of traditional RNN architectures in handling complex grammatical structures.

An interesting point of comparison is the reported limitation of stack-RNN memory size in learning grammars, which Mali et al. [40] noted was in line with automata theory. This observation provides additional context for our investigation into the memory requirements of neural networks for grammar learning tasks.

The NSPDA study focused specifically on CFGs and employed a two-stage incremental learning procedure along with an adaptive noise regularization scheme. In contrast, our research extends to a broader range of grammatical structures and uses different learning algorithms. Despite these methodological differences, the success of the NSPDA in recognizing CFGs, particularly its ability to generalize to sequences longer than those in the training set, supports our hypothesis about the importance of external memory structures in enhancing the grammatical learning capabilities of neural networks.

This comparison with the NSPDA work reinforces the potential of memory-augmented neural networks in tackling complex grammatical tasks and provides a valuable benchmark for our own investigations into neural models of cognitive functions.

6.1.3 Neural Networks and the Chomsky Hierarchy

Delétang et al. [32] conducted a comprehensive empirical study on the performance of various neural network architectures across different levels of the Chomsky hierarchy. Their work shares significant methodological similarities with this thesis, particularly in the use of artificial grammars to test the computational capabilities of neural networks. Both studies employ this approach to allow for a controlled comparison across different levels of the Chomsky hierarchy, providing insights into the generalization capabilities of various network architectures.

While Delétang et al. [32] focused primarily on machine learning and artificial intelligence aspects without exploring cognitive science implications, the present research extends these findings to cognitive modeling. This distinction highlights the value added by this thesis in bridging AI research and cognitive science. When combined with the present research, their methodology and findings have significant relevance to understanding cognitive functions.

The study by Delétang et al. [32] assessed the generalization capabilities of networks by testing on sequence lengths considerably larger than those used in training, an approach similar to the one used in this thesis. Their study examined a range of architectures including RNNs, LSTMs, transformers, and memory-augmented networks such as Stack-RNNs and Tape-RNNs. Of these, Tape-RNNs can be considered analogous to the Neural Turing Machines (NTMs) used in this study, as both feature an indexable memory.

Their key findings align well with the results of this thesis:

- RNNs generally solved tasks up to the regular level of the Chomsky hierarchy.
- LSTMs performed well on regular tasks and some counter-language tasks, supporting the findings of this study on the A^nB^n language.
- Stack-RNNs could solve up to deterministic context-free tasks, which aligns with the findings of Mali et al. [40].

- Tape-RNNs demonstrated capability in solving context-sensitive tasks, though not all of them. This finding parallels the observations of NTM capabilities in context-free and mildly context-sensitive tasks in this study.
- Transformers struggled with many regular tasks due to issues with positional encodings, highlighting the importance of architecture design in task performance.

The study by Delétang et al. and this thesis both emphasize the importance of memory mechanisms in neural network architectures. The performance of Stack-RNNs and Tape-RNNs in their study, and NTMs in this thesis, underscores the crucial role of external memory in infering the rules at higher levels of the Chomsky hierarchy. This aligns with cognitive science theories about the role of working memory in syntactic processing and other complex cognitive tasks. The ability of these memory-augmented networks to handle context-sensitive tasks suggests potential parallels with human cognitive processes that require maintaining and manipulating complex contextual information.

Importantly, their study reinforces the findings of this thesis on the significance of inductive biases in architectures for solving different classes of problems. In particular, it underscores the crucial role of memory mechanisms in generalizing tasks across different levels of the Chomsky hierarchy, a central theme in this investigation of cognitive functions through neurocomputational models.

6.1.3.1 Further Complexity Analysis with MCSGs

While Delétang et al. [32] did not directly explore cognitive science implications, their work, when combined with the present research, offers valuable insights into the computational requirements for different levels of syntactic processing in the brain. Of particular interest are the context-sensitive tasks in their study, which can be analyzed in terms of their mildly context-sensitive nature. However it should be noted that these analyses are made based on conditions described in Section 3.3.6.1. Grammars are evaluated according to these properties since we do not have a mathematical proof.

Mildly context-sensitive grammars (MCSGs) are of special interest in cognitive science due to their ability to capture many of the complex structures found in natural languages while remaining computationally tractable. The properties of MCSGs, as described in Section 3.3.6.1, provide a framework for analyzing the tasks in Delétang et al.'s work.

Mildly Context-Sensitive Tasks

Several of the context-sensitive tasks in Delétang et al.'s study can be classified as mildly context-sensitive:

- **Duplicate String**: This is a copy language, which is known to be mildly context-sensitive. Copy languages extend beyond the capabilities of context-free grammars while maintaining desirable properties such as polynomial parsing time. Controlled copying operations fit within the mildly context-sensitive framework [160].
- **Missing Duplicate**: Similar to the Duplicate String task but with one token missing, this task shares the mildly context-sensitive properties of its counterpart.

- **Odds First**: Another copying task with controlled conditions, this can also be considered mildly context-sensitive. Research indicates that structured copying operations do not exceed the boundaries of MCSLs [175].
- **Binary Addition**: While this task involves a carrying operation that creates dependencies beyond context-free grammars, its limited cross-dependencies qualify it as mildly context-sensitive.
- **Bucket Sort**: This task has limited dependencies (counting occurrences), can be parsed in linear time with a parallel counting strategy, and exhibits constant growth. While context-sensitive, it is not context-free, thus qualifying as mildly context-sensitive.

All of these tasks exhibit the key properties of MCSGs: limited crossing dependencies, polynomial-time parsing (often linear), and constant growth.

Beyond Mildly Context-Sensitive Tasks

Some tasks in Delétang et al.'s study likely exceed the bounds of mildly context-sensitive grammars:

- **Binary Multiplication**: This task involves more complex dependencies than addition and may have unlimited crossing dependencies depending on implementation. It requires quadratic parsing time and may not exhibit constant growth, likely disqualifying it as an MCSG.
- **Compute Sqrt**: This task involves complex dependencies, may require more than polynomial time for parsing, and may not exhibit constant growth, suggesting it is beyond the scope of MCSGs.

Tape-RNN Performance and Mildly Context-Sensitive Grammars

An analysis of the Tape-RNN's performance on these tasks reveals a strong correlation with the mildly context-sensitive nature of the grammars:

- **Successful Generalization**: The Tape-RNN was able to fully generalize on the Duplicate String, Missing Duplicate, Odds First, and Binary Addition tasks, all of which are classified as mildly context-sensitive.
- **Failure to Generalize**: The Tape-RNN failed to generalize on the Binary Multiplication and Compute Sqrt tasks, which are likely beyond mildly context-sensitive.
- **Partial Success**: On the Bucket Sort task, which is mildly context-sensitive but permutation-invariant, the Tape-RNN showed significant success but did not fully generalize.

This pattern of Tape-RNN performance strongly correlates with the mildly context-sensitive nature of the tasks. The Tape-RNN's perfect generalization on MCSGs and struggle with more complex tasks mirrors the hypothesized capabilities of human language processing.

6.1.4 Neural Turing Machines

Graves et al. [38] introduced Neural Turing Machines (NTMs) as a novel neural network architecture that extends the capabilities of standard recurrent neural networks (RNNs) by incorporating a differentiable memory matrix. This architecture is designed to mimic the operation of a Turing machine, providing a neural network with an external memory that can be read from and written to via attention-based mechanisms. This addition allows NTMs to learn and execute simple algorithms, making them a powerful tool for solving a wide range of sequential problems.

While Graves et al. [38] concentrated on the artificial intelligence and machine learning dimensions of Neural Turing Machines (NTMs) without addressing their cognitive science implications, this thesis expands over their work by applying it to cognitive modeling. By incorporating the computational strengths of NTMs into cognitive frameworks, this thesis aims to offer deeper insights into the neuro-computational mechanisms underlying human cognition, thereby extending the relevance and application of NTMs beyond their original scope.

Graves et al. [38] conducted several experiments to demonstrate the capabilities of NTMs, including tasks such as copying sequences, associative recall, and sorting. These tasks, although not explicitly framed within the context of the Chomsky hierarchy and cognitive science, provide valuable insights into the potential of NTMs to handle complex structured data:

- **Copy Task (WW Grammar)** : NTMs demonstrated the ability to store and recall long sequences of arbitrary information, outperforming LSTMs in terms of learning speed and accuracy. The NTM generalized almost perfectly to longer sequences than those seen during training, with only minor errors on the longest sequences. This task is related to the recognition of regular languages (Type 3 in the Chomsky hierarchy).

- **Repeat Copy Task (WWW^* Grammar)**: The NTM effectively generalized to both longer sequences and more repetitions than seen during training. Although it occasionally struggled with counting the number of repeats correctly when they exceeded the training range, its overall performance indicates strong generalization capabilities. This task also relates to regular languages.

- **Associative Recall**: NTMs successfully learned to retrieve subsequent items in a sequence based on a given query item. They maintained performance for sequences up to twice the length of those used in training, reflecting the network's ability to manage structured memory. This task aligns with context-free languages (Type 2).

- **Priority Sort**: NTMs sorted binary vectors based on assigned priorities, demonstrating an understanding of complex relationships and dependencies characteristic of context-sensitive languages (Type 1). The NTM achieved significant generalization capabilities, although not perfect, indicating the potential to handle mildly context-sensitive grammars.

The findings of Graves et al. align well with the observations in this thesis regarding the performance of NTMs on various artificial grammar tasks:

- NTMs can solve tasks up to the context-sensitive level of the Chomsky hierarchy. Their external memory mechanism enables them to manage dependencies and hierarchical structures more effectively than standard RNNs and LSTMs.

- The capability of NTMs to generalize tasks across different lengths and complexities, as shown in the copy and associative recall tasks, parallels the ability to handle context-free and mildly context-sensitive languages observed in the experiments conducted in this thesis.

- The success of NTMs in sorting tasks emphasizes the importance of memory mechanisms in neural network architectures for solving higher-order language tasks, reinforcing the central theme of this research on the role of memory in cognitive functions.

6.1.4.1 Complexity Analysis Using Chomsky Hierarchy and MCSGs

The experiments conducted by Graves et al. [38] provide a foundation for analyzing the computational requirements of different levels of syntactic processing in the brain. Their work highlights the significance of memory-augmented neural networks in modeling complex cognitive tasks, particularly those involving hierarchical and context-sensitive information. As explained in Section 6.1.3.1, MCSG category is decided by their properties since we do not have a formal mathematical proof.

Mildly Context-Sensitive Tasks

Several tasks explored by Graves et al. [38] can be classified as mildly context-sensitive, offering a framework for understanding their relevance to natural language processing and cognitive modeling:

- **Associative Recall**: This task involves indirection, where one data item points to another, mirroring the nested structures in natural languages that are captured by mildly context-sensitive grammars (MCSGs). NTMs generalized perfectly in this task.

- **Priority Sort**: Sorting based on priorities requires managing dependencies and ordering relationships, aligning with the properties of MCSGs that extend beyond context-free grammars while remaining computationally tractable. NTMs achieved significant but not perfect generalization.

- **Copy Task**: The ability to store and recall long sequences of arbitrary information demonstrates handling of sequential dependencies, fitting within the MCSG framework. NTMs generalized perfectly in this task.

- **Repeat Copy Task**: This task involves controlled repetition, which aligns with the properties of MCSGs. NTMs generalized effectively but struggled with very high repetitions.

The performance of NTMs on these tasks supports their potential application in cognitive science for modeling human syntactic processing capabilities, particularly in tasks requiring the maintenance and manipulation of complex contextual information.

Beyond Mildly Context-Sensitive Tasks

While NTMs exhibit strong performance on mildly context-sensitive tasks, certain tasks explored by Graves et al. may exceed the capabilities of MCSGs:

- **Dynamic N-Grams**: This task tests the ability to adapt to new predictive distributions, involving complex statistical dependencies that may challenge the boundaries of MCSGs.
- **Binary Multiplication**: The dependencies involved in this task likely surpass the scope of MCSGs, requiring more intricate computational mechanisms.

NTM Performance and Mildly Context-Sensitive Grammars

An analysis of NTMs' performance on these tasks reveals a strong correlation with the mildly context-sensitive nature of the grammars:

- **Successful Generalization**: NTMs effectively generalized on tasks like associative recall and priority sort, which align with the properties of mildly context-sensitive grammars.
- **Failure to Generalize**: NTMs struggled with tasks like dynamic n-grams and binary multiplication, which likely exceed the computational capabilities of MCSGs.

This pattern of performance underscores the potential of NTMs to model some human cognitive processes, particularly those involving hierarchical and context-sensitive information processing, thereby contributing to the understanding of the computational mechanisms underlying cognitive functions.

6.2 Language and Hierarchical Processing Implications Related to Comparative Studies

The results of our study on artificial grammar learning in neural networks have significant implications for our understanding of language processing in both humans and animals. These findings align well with existing research on comparative linguistics and cognitive neuroscience.

6.2.1 Animal Limitations and Human Capabilities

Our results are aligned with the theory that non-human animals are limited to processing regular grammars, due to their lack of a dorsal pathway connection. This connection is crucial for syntactic processing and working memory as discussed in Section 2.3.1. From a computational perspective our results underscore the importance of control and memory mechanisms for syntactic processing. The superior performance of Neural Turing Machines (NTMs) in learning and generalizing supra-regular grammars suggests that one of the main factors distinguishing human syntactic abilities might arise from the ability to use an addressable memory that can store spatio-temporal relevance. This aligns with the observed differences between humans and other primates in the processing of complex grammatical structures [97, 101].

Moreover, there are parallels between humans capacity and limitations on learning mildly context-sensitive grammars in language and NTM's performance is notable and will be discussed in detail in Section 6.3.2

6.2.2 Recurrent Memory and Sequence Length

The performance of Long Short-Term Memory (LSTM) networks in our experiments provide insights into the role of recurrent memory in syntactic processing. LSTMs, while proficient at regular grammars and some counting tasks, were unable to generalize most supra-regular grammars. However, their performance on complex grammars being limited to short sequence lengths, suggests that the importance of specialized memory mechanisms increases with the length of the sequences. This observation is consistent with the findings reported in Section 2.3.1, where animals showed diminishing ability to process context-free structures as the sequence length increased [94].

Our findings also align well with neuroscientific evidence related to possible involvement of the hippocampus explained in Section 2.4.5.3, particularly the case of H.M., a patient with severe amnesia following bilateral medial temporal lobectomy. H.M. was reported to maintain his syntactic abilities [136], but struggled with longer sequences [137]. This pattern of preserved basic language skills coupled with impaired processing of more complex structures may suggest the importance of specialized memory structures in advanced language processing. However, the results are suggestive rather than conclusive and require further investigation.

6.2.3 Syntactic Complexity and Cognitive Simulation Framework

The cognitive simulation framework developed in this study provides a valuable tool for testing theories of syntactic processing at an algorithmic level. The results are generally compatible with research findings from behavioral and neuroscientific studies explained in Section 6.7 which highlights the importance of the connection between syntactic processing and working memory. As our understanding of language, syntactic processing and memory mechanisms grows, it will be possible to refine the models, tasks, and memory access mechanisms accordingly.

This framework offers a bridge between computational models and empirical observations, allowing for more nuanced investigations into the cognitive mechanisms underlying language processing. By simulating various memory architectures and their impact on grammar learning, we can generate testable hypotheses about the neural substrates of language in humans and other species.

Future work could focus on further refining the memory mechanisms in our models to more closely mimic the observed capabilities and limitations of different species. Additionally, expanding the range of grammatical structures tested could provide even more detailed insights into the relationship between memory architecture and syntactic capabilities.

6.3 Cognitive Science Implications of Memory Networks

This section studies the implications of memory networks in artificial grammar learning tasks. We will discuss the effects of the memory used in a neural network architecture on the complexity of the tasks it can perform and grammars that it can learn.

The similarities between the human mind and capabilities of augmented memory networks will be discussed from an algorithmic and structural perspective. Similar to humans, their ability to learn mildly context sensitive grammars and their limitation to go beyond that complexity will be discussed.

6.3.1 Neural Network Architectures and Grammatical Complexity

The grammatical complexity of sequences, as defined by the Chomsky hierarchy, plays a crucial role in understanding the capabilities of neural networks in processing language. In terms of grammatical complexity, human language is generally considered to have mildly context-sensitive properties, exhibiting mostly context-free properties with limited context-sensitive features [175]. This results in a predominantly tree-like structure with some crossing dependencies [176], as explained in Section 3.3.6.1. While recurrent neural networks (RNNs) are theoretically universal function approximators [177, 178], practical limitations prevent them from approximating every function effectively. This is evidenced by varying success rates across different grammar and architecture combinations, as shown in Section 4.3. Experiments conducted in this study (Section 4.3) and related work (Section 6.1) support several key findings:

- RNNs without external memory mechanisms generally fail to generalize beyond regular grammars, except for those resolvable by counting.
- Consistent with automata theory, neural networks require stack-like (or more capable) memory mechanisms to generalize context-free grammars effectively. Even though it's theoretically possible for RNNs to approximate any function [177, 178], experiments show that this capability cannot be learned through training with examples.
- Memory-augmented neural networks can generalize most context-free grammar tasks and some context-sensitive grammar tasks.
- A common property of context-sensitive languages successfully generalized by memory-augmented networks is mild context-sensitiveness.

These observations underscore the importance of aligning neural network architectures with the specific grammatical complexities they aim to process, which mirrors the relationship between grammar types and corresponding automata in formal language theory. Similar to formal language theory, neural networks require stack-like memory mechanisms to process CFGs or mechanisms similar to finite tape are required for CSGs. But our results show that this memory mechanism is not directly the predictor of this capacity but acts more as a prerequisite.

6.3.2 Memory Networks' Ability to Generalize Mildly Context-Sensitive Tasks

Memory networks, particularly Neural Turing Machines (NTMs) and Tape-RNNs, have shown impressive abilities to generalize tasks that match Mildly Context-Sensitive Grammars (MCSGs). This section looks at how these networks perform on various tasks, grouping them based on their MCSG status and examining the link between a task's MCSG classification and how well networks can generalize it.

One of the experiments performed as part of this thesis involved the AWA ($WW^R\_WW^R$) grammar, as shown in Section 5.2. This grammar was specifically designed as a Combinatory Categorial Grammar (CCG) to be an MCSG that combines a mirror (WW^R) grammar with repetition(W_W). NTMs successfully generalized this grammar, which is especially important because it was designed to be both complete and sound when generalized, as explained in Section 3.3.8.5.

Another experiment performed as a part of this thesis was the AWA-Star ($WW^R\_WW^R(\_WW^R)^*$) grammar. It was designed to be slightly more complex than the AWA (WW^R) grammar by following with many repetitions of the same word. The results are detailed in Chapter 5.

As discussed in Section 6.1.4.1, NTMs [38] also successfully generalized several other MCSG tasks, including:

- Associative Recall
- Copy Task (WW Grammar)
- Repeat Copy Task
- Priority Sort (with significant but not perfect generalization)

Similarly, Tape-RNNs, as detailed in Section 6.1.3.1, showed skill in generalizing MCSG tasks such as:

- Duplicate String (WW Grammar)
- Missing Duplicate
- Odds First
- Binary Addition

However, both NTMs and Tape-RNNs had trouble with tasks that likely go beyond MCSG complexity, like Binary Multiplication and Compute Square Root. Interestingly, Bucket Sort, despite being an MCSG, was not successfully generalized, standing out as an exception.

Table 9 summarizes these tasks, their MCSG status, and whether memory networks could generalize them.

Table 9: Grammar tasks, MCSG classification, and generalization by memory networks

Task	**MCSG**	**Generalized**
AWA (WW^R_WW^R) Gr.	Yes	Yes
AWA-Star (WW^R_WW^R(_WW^R)*) Gr.	Yes	Yes
Associative Recall	Yes	Yes
Priority Sort	**Yes**	**Significant**
Copy Task (WW Grammar)	Yes	Yes
Repeat Copy Task	Yes	Yes
Duplicate String (WW Grammar)	Yes	Yes
Missing Duplicate	Yes	Yes
Odds First	Yes	Yes
Binary Addition	Yes	Yes
Bucket Sort	**Yes**	**No**
Dynamic N-Grams	Yes	No
Binary Multiplication	No	No
Compute Square Root	No	No

The results show a strong link between a task's MCSG classification and the ability of memory networks to generalize it. Most MCSG tasks were successfully generalized by both NTMs and Tape-RNNs, with Bucket Sort being a notable exception. Priority Sort achieved significant, though not perfect, generalization. The successful generalization of the WW^R_WW^R (AWA) grammar by NTMs, as part of this thesis, further strengthens this correlation, providing a carefully designed example of MCSG generalization. In contrast, CSG tasks beyond MCSG were not successfully generalized.

The Bucket Sort exception and the partial success with Priority Sort highlight that, while MCSG classification is a good indicator, other factors can also affect a network's ability to generalize certain tasks. It is important to note that the MCSG nature of grammars is determined by analyzing certain properties rather than through formal proofs.

This pattern suggests that MCSG classification is a reliable predictor of whether memory networks can generalize a task, but with some exceptions. The success of memory networks with MCSG tasks, especially the specifically designed AWA grammar, likely comes from their ability to handle limited cross-dependencies and show constant growth, which are key features of MCSGs. These abilities allow memory networks to capture and generalize the structured patterns in most MCSG tasks, similar to syntactic complexities and limitations of human language.

The strong tendency of memory networks, especially NTMs, to generalize MCSG tasks while struggling with more complex, non-MCSG tasks suggests that MCSG properties are fundamental to generalization in both artificial and biological neural systems. This ability to process complex, hierarchical structures is similar to what we see in human languages.

In summary, the existence of an external addressable memory is generally a prerequisite for generalizing most of the CSGs. Although for memory-augmented neural networks, prediction of generalization based on MCSG classification is not absolute, it is highly indicative. The successful generalization of the AWA grammar, designed as a CCG to be both complete and sound, provides strong evidence for

this relationship. This connection between MCSGs and neural network generalization offers valuable insights into how both artificial and biological systems might process complex language structures.

6.3.2.1 Important Properties of MCSGs for Performance of Addressable Memory Networks

The following properties of the tasks-architecture combination are presented to be important properties for performance and generalization in inductive learning tasks.

1. **Limited Cross-Dependencies**: MCSGs have limited crossing dependencies, which align well with the memory network's ability to manage information through structured memory mechanisms.
2. **Polynomial-Time Parsing**: The polynomial-time (often linear) parsing characteristic of MCSGs matches the computational efficiency of the memory network's operations.
3. **Constant Growth Property**: This property of MCSGs may allow memory networks to develop generalizable strategies during training that scale to longer sequences.
4. **Structured Memory Access**: The ability of memory networks to read and write to specific memory locations could be particularly suited to the structured nature of mildly context-sensitive tasks.

The success of memory networks, such as NTMs, on tasks like associative recall and priority sort, which align with the properties of MCSGs, suggests that these architectures can adapt to complex, structured tasks. However, the lack of full generalization on tasks like dynamic n-grams indicates that there are limits to their capabilities with more complex dependencies.

Specifically, the Tape-RNN's partial success on the Bucket Sort task, despite its permutation-invariant nature, suggests that this architecture can adapt to some degree to tasks that don't require strict sequential processing. However, the lack of full generalization indicates that permutation-invariance may pose additional challenges to the memory networks for learning these tasksç

Conversely, the Tape-RNN's failure on tasks beyond MCSGs (Binary Multiplication and Compute Sqrt) could be due to:

- The potential for unlimited crossing dependencies, which may exceed the Tape-RNN's capacity to manage information effectively.
- The higher time complexity of these tasks, which may require computational resources beyond what the Tape-RNN can provide within a reasonable number of steps.
- The possible lack of constant growth property, which could hinder the Tape-RNN's ability to develop scalable strategies.

This distinction in performance between mildly context-sensitive tasks and those beyond provides compelling evidence for the cognitive plausibility of such architectures for syntactic tasks. It insicates

that the computational properties of MCSGs might be fundamental to both artificial and biological neural systems capable of processing complex, hierarchical structures like those found in human languages.

6.4 Why Memory Networks Can Generalize Mildly Context-Sensitive Grammars

Previous sections have demonstrated that memory networks such as Neural Turing Machines (NTMs) and Tape-RNNs are capable of learning grammars up to the complexity of Mildly Context-Sensitive Grammars (MCSGs). Notably, these networks could not generalize context-sensitive grammars more complex than MCSGs. This section explores the factors that may have contributed to these results.

6.4.1 Structural Bias Provided by Addressing Mechanisms

The addressing mechanism in memory networks provides a crucial structural bias that enables the handling of complex dependencies:

- It allows token representations to be saved and recalled efficiently.
- The mechanism can access the correct address and retrieve information as needed.
- Importantly, temporal relations of tokens are preserved in the addressing space.

Comparing this to other architectures:

- Recurrent Neural Networks (RNNs) can theoretically access the same information, but lack an inherent structural bias to order elements in sequence by space or time.
- Networks like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have additional mechanisms, but these are insufficient to inherently preserve positional information.
- Transformers can attend to all input tokens directly and use positional encoding to preserve spatiotemporal relations [54]. However, work by Delétang et al. [32] showed that transformers could not generalize the problem of duplication, suggesting this strategy is insufficient for generalization.
- The Stack-RNN structure in Delétang et al.'s work [32] provides a stack mechanism allowing playback of representations in reverse order, making it successful at reversing strings but not duplicating them.

Memory networks like NTMs or Tape-RNNs provide an addressable space to store and recall representations. This creates an important **structural bias** in the network which inherently preserves the temporal or spatial relations of the tokens. This is analogous to how Convolutional Neural Networks (CNNs) benefit from a structural bias towards spatially close items, greatly improving performance and learnability in visual tasks.

We suggest that this structural bias towards maintaining temporal or spatial relations between representations significantly enhances the performance and learnability of the problem, enabling generalization of MCSGs. This bias is reminiscent of spatiotemporal relations in hippocampal place cells and/or the phonological loop in the human brain, indicating that the connection between language processing areas and memory areas capable of maintaining spatiotemporal relations may be an important factor for humans' capacity for learning and processing complex languages like MCSGs. This is supported by some neuroscientific evidence. Hippocampus is known to provide a neural basis for keeping spatial and temporal relations via the place and time cells [130]. Hippocampus is also known to be related to interaction with Broca's area and encoding of syntactic structures [110].

6.4.2 Computational Complexity of MCSGs

The polynomial parsing characteristics and constant growth property of MCSGs limit the computational complexity of the system. When the algorithmic complexity of a problem far exceeds linear complexity, the required computational steps increase significantly. This situation:

- Raises control issues where the number of computational steps or time taken should be much higher than for the input sequence. Additional computations are required, and internal representations or mechanisms must be updated to reach a conclusion. This can be observed in various computing systems:
  - Tape and internal state updates in Turing machines
  - Memory and register updates in digital computers
  - Recurrent state activation updates in RNNs
  - External memory updates in memory-augmented neural networks
- Can complicate the learning scenario as the amount of computation and activations increase (since each output results from many steps), making it more challenging for any learning system to propagate error and update the system effectively.

The ability of memory networks to handle the computational complexity of MCSGs, while struggling with more complex grammars, aligns with these observations. The addressable memory and controlled access mechanisms in NTMs and Tape-RNNs provide an effective balance between computational power and learnable structure, well-suited to the characteristics of MCSGs.

6.4.3 Independence of Memory from Compute

A fundamental distinction between architectures like LSTM (or a classical RNN) and NTM lies in their approach to memory management. While LSTMs improve upon classical RNNs by offering better control and separation of memory, the memory component remains integral to the processing mechanism [139]. In contrast, NTMs provide a clearer separation through differentiable external memory access mechanisms.

This separation of compute and memory is reminiscent of classical computing architectures such as the von Neumann architecture or formal descriptions such as Turing machine and linearly bounded

automata, where such distinction offers several advantages. Turing machines, for instance, operate with a finite number of states but with an infinite tape as memory, proving capable of computing anything computable with finite-state computing mechanisms.

The von Neumann architecture similarly demonstrates this separation, with the processing unit operating on a finite set of instructions while the system as a whole can manage a vast number of states. This architectural choice provides an effective strategy for managing computational complexity: the controller executes a finite set of instructions, while the memory stores the vast amount of data and intermediate results.

An illustrative example from classical computer architectures is the distinction between RISC (Reduced Instruction Set Computing) and CISC (Complex Instruction Set Computing). CISC architectures, with their more complex instructions, can perform more operations in a single step but at the cost of increased design complexity and operational expense. Conversely, RISC architectures, with their simpler and fewer instructions, can execute these instructions more rapidly. From a system design perspective, the simpler computing architecture of RISC has often proven more efficient and versatile within the von Neumann paradigm [179, 180, 181].

These observations support the notion that for systems performing symbolic computation, the separation of memory and computation can confer significant advantages. This separation allows the controller system to be more flexible and adaptable. In the context of neural systems learning from data, this distinction may be a crucial factor explaining why RNNs, despite their theoretical capability to approximate any function, struggle to learn the rules of formal grammatical tasks effectively.

While this perspective on the separation of compute and memory is not yet fully substantiated by psychological or neuroscientific evidence, it offers distinctive advantages from an information processing and computing standpoint. As such, it represents an important factor to consider when seeking to understand both artificial and biological neural systems.

6.5 Bridging the Gap Between Connectionist and Symbolic Processing

The proposed framework can help in the aim to bridge the gap between symbolic and sub-symbolic approaches to cognition. By demonstrating how neural networks can learn and generalize grammatical rules, it might contribute to the ongoing debate about the nature of cognitive representations. Researchers could use this framework to investigate how symbolic-like processing emerges from the sub-symbolic operations of neural networks, potentially offering a unifying perspective on these two approaches to cognition.

Previous works on neurosymbolic processing have often used hybrid architectures that combined symbolic and connectionist parts. This allows them to take advantage of the most relevant parts of each approach, but does not fully explain how symbol processing is possible in a purely connectionist framework.

Several properties of symbolic systems and their similarities with this framework are worth noting [182, 183]:

6.5.1 Symbolic Representations

Symbolic representations are discrete symbols that stand for objects, concepts, or relationships. Symbolic processing must be precise, discrete and abstract [182]. Unlike sensory representations, which are continuous and tied to specific modalities (e.g., visual images or sounds), symbolic representations are discrete entities that can be combined and manipulated according to rules.

The explicitness of symbolic representations requires them to be recalled near perfectly rather than loosely. The memory of classical RNNs without external memory structures (Vanilla RNN, LSTM, GRU) uses the recurrent memory structures to store the representations. Gate structures can decide when to store or remove representations, but the whole memory matrix is still updated. Since there are no exclusive memory slots, with each update, previous information is susceptible to degradation, indicating loosely recalled elements rather than discrete ones.

In contrast, memory networks such as NTMs can store memory in different slots without affecting others. *This allows the representations to be discrete and precise as required by symbolic systems.*

6.5.2 Compositionality

Symbolic representations can be combined to form complex structures that represent more complex ideas. For example, the words "cat" and "sat" can be combined to form the sentence "The cat sat," representing a specific event. This compositionality allows us to generate an infinite number of thoughts from a finite set of symbols.

This property requires representations to be in the same representational space. These representations should be consistent in a way that they can be combined. Usually when something is processed in a layer of neural network it is passed to a different group of neurons which has a different representational space.

Having retrievable memory mechanisms allows for a common representational space that can store representations in the same space similar to symbolic processing. For example, in a binary representation, a vector of [0 0 1 0] represents the number 2 in decimal space and [0 0 1 1] represents the number 3. When we apply the addition operation with [0 0 1 1] and [0 0 1 0] the result is [0 1 0 1] which is the number 5. *The representations provide a consistency throughout operations which is one of the prerequisites of being a symbol.*

6.5.3 Rule-governed

The manipulation of symbolic representations is governed by rules such as the rules of grammar or logic. These rules determine how symbols can be combined and transformed to create new meanings. This rule-governed nature allows us to reason systematically and draw inferences.

The proposed framework focuses on generalization. As explained in Section 3.3.8, the learning scheme is very strict. It ensures the system can understand all possible outputs, which is possible by learning the rules. The framework tests the systems' capacity to generalize beyond training examples by learning the rules.

When a model achieves generalization with 0 cost in the framework, this demonstrates the capacity to apply the rules in the grammar accordingly. **With this we can consider the generalized models to be work as rule-governed as required by symbolic systems.**

6.5.4 Serial Processing

Symbolic cognitive models typically operate in a step-by-step, sequential manner, mirroring the logical progression of human reasoning and syntax processing.

The framework focuses on processing tokens sequentially. As explained in Section 2.5.3, at each given input, the system updates the memory. In the context of this experimental framework, the outputs are possible tokens.

This allows the processing of the tokens sequentially in a serial way when compared to architectures like CNNs or transformers. As a limitation of the framework, these properties only allow the system to compute the results when the computation required scales linearly with the input. But a symbolic system would be able to handle higher complexities.

This limitation of sequential models is known in the literature and solutions are proposed. One notable solution by Graves et. al, is to dynamically decide the number of computational steps to take between receiving an input and producing an output [184].

6.6 Discussion on the Use of Legendre Memory Units (LMUs) in Experiments

6.6.1 Motivations for Selecting LMUs

Legendre Memory Units (LMUs) were selected for this study due to their unique memory mechanism and their potential for providing a more biologically plausible model of memory processes. LMUs use orthogonal polynomial functions to maintain a continuous representation of the input history, which aligns more closely with how biological neurons process temporal information. This characteristic, combined with their computational efficiency and the potential for seamless integration of temporal data, made LMUs a compelling candidate for exploring their capabilities in learning and generalizing various types of artificial grammars.

6.6.2 Memory Mechanism Differences

LMUs employ orthogonal polynomial functions, specifically Legendre polynomials, to compress and maintain a continuous representation of the input history. This implicit memory update mechanism relies on the mathematical properties of these polynomials to integrate temporal information seamlessly over time. This continuous representation contrasts with the explicit gating mechanisms in LSTMs, where information flow into and out of the cell state is controlled through gates, enabling precise and targeted memory updates.

6.6.3 Impact on Performance

The distinct memory mechanisms of LMUs and LSTMs led to different performance outcomes across the tasks. LMUs demonstrated strong generalization capabilities in tasks involving regular grammars. Their ability to maintain a continuous representation of temporal information proved advantageous in these scenarios, as it allowed for a holistic integration of the sequence data.

However, in tasks such as the A_nB_n grammar, which require precise memory operations like counting and balancing sequences, LSTMs outperformed LMUs. The explicit gating mechanism of LSTMs provided the necessary control to count and match tokens effectively, which is critical for accurately processing AnBn sequences. In contrast, the implicit updates in LMUs, while efficient for continuous integration, lacked the precision required for exact counting tasks, leading to less effective generalization in these cases.

6.6.4 Generalization and Memory Updates

The difference in generalization performance between LSTMs and LMUs, particularly in the A_nB_n grammar task, can be attributed to their respective memory update mechanisms. LSTMs can selectively retain and forget specific pieces of information through their gates, enabling them to count the number of 'A' tokens and match them with 'B' tokens precisely. This ability to manage memory explicitly allows LSTMs to generalize effectively by leveraging their targeted memory control.

On the other hand, LMUs, with their implicit memory updates, manage memory through a continuous and compressed representation of past data. This approach is beneficial for tasks requiring a smooth integration of temporal information but may not support exact operations such as counting. The inability to explicitly control memory updates could lead to challenges in tasks that demand precise and discrete memory manipulations.

6.6.5 Implications for Cognitive Modeling

The findings from these experiments suggest that while LMUs offer a computationally efficient and biologically plausible memory mechanism, their performance may vary based on the nature of the task. LMUs excel in scenarios requiring continuous integration of temporal information, making them suitable for tasks where a holistic temporal context is more critical than precise memory operations. Conversely, tasks necessitating exact memory control, such as counting or balancing sequences, benefit more from the explicit gating mechanisms of LSTMs.

This distinction allows the selection of a different variety of neural network architectures tailored to the specific cognitive tasks or cognitive process being modeled. LMUs, with their biologically inspired design, provide valuable insights into continuous memory integration, while LSTMs offer robust solutions for tasks requiring precise memory control.

In conclusion, the use of LMUs in these experiments highlights their potential and challenges in cognitive modeling. Their continuous and efficient memory representation is a significant advantage for certain tasks, yet their implicit memory updates may limit their performance in tasks requiring precise memory manipulations. This study contributes to the broader understanding of how different neural

network architectures can be applied to model various aspects of human cognition, paving the way for more refined and effective computational models.

6.7 Neuroscientific Connections

The findings from our computational models of memory networks reveal intriguing parallels with human neuroscientific systems, particularly in the domain of language processing and syntactic capabilities. These similarities not only validate our approach but also offer insights into the potential neural mechanisms underlying complex cognitive functions.

6.7.1 Syntactic Processing and Memory

Our research highlights a striking similarity between the performance of Neural Turing Machines (NTMs) and human syntactic processing capabilities. Both demonstrate proficiency up to the level of Mildly Context-Sensitive Grammars (MCSGs), suggesting a fundamental computational limit that may be shared between artificial and biological systems. This parallel is particularly noteworthy when considering the neural basis of human language processing.

In humans, syntactic processing primarily relies on Broca's area and its connections with the temporal lobe. Neuroscientific evidence has long suggested a strong relationship between working memory and Broca's area [110]. Our computational models, particularly the NTM architecture, seem to mirror this relationship, indicating that the integration of working memory-like mechanisms is crucial for processing complex grammatical structures.

6.7.2 Memory Mechanisms: Artificial and Biological

Our findings suggest that generalizing more complex grammars requires an external memory mechanism, a feature prominently displayed by NTMs. This observation aligns with neuroscientific theories about the role of specialized memory systems in language processing. The memory used in our neural networks, particularly in NTMs, shares structural and functional similarities with human working memory:

Both are not necessarily long term and transient and for both cases information is stored in activations rather than permanent synaptic changes. Both mechanisms are capable of re-activating previous representations of distinct concepts.

The recurrent memory in LSTMs and LMUs can be seen as analogous to cortical memory or the recurrent activation patterns in Broca's area. This aligns well with their good performance on regular grammar tasks and sufficient performance with shorter sequences of more complex grammars. Interestingly, this performance profile shows parallels with the grammatical capabilities of macaque monkeys, as discussed in Section 2.3.1.

6.7.3 NTMs and Human Memory Systems

Although they are far from being a complex model of human memory, the NTM architecture, with its external and separate memory mechanism, offers particularly compelling parallels to specialized memory mechanisms in the human brain. The two types of memory access in NTMs find analogues in human memory systems:

- **Content-addressable memory** in NTMs bears similarities to semantic memory in humans. Both systems record relationships between different pieces of information, allowing for retrieval based on content rather than specific addresses. This enables both NTMs and humans to link concepts and facts, facilitating understanding and reasoning based on the meaning of information rather than its location.
- **Location-based memory** in NTMs parallels episodic memory in humans. Both systems store and retrieve information based on the specific context or sequence in which the information was encountered. This allows for the recall of events and the order in which they occurred, making it possible to reconstruct timelines and experiences.

The need for memories to be stored in a sequentially ordered and addressable way in NTMs might suggest a potential analogue to the role of the hippocampus in human memory. The hippocampus is crucial for the formation and retrieval of episodic memories and plays a significant role in spatial and temporal ordering of information [130].

These parallels between our computational models and the neuroscientific understanding of human memory and language processing can provide a promising avenue for further research. They suggest that our models may be capturing fundamental computational principles that underlie complex cognitive functions, offering a bridge between artificial intelligence and cognitive neuroscience.

6.8 Computational Modeling and Simulation Framework

The computational modeling and simulation framework developed in this study offers a versatile platform for investigating various neural network architectures and their applications in modeling cognitive functions. This section discusses the framework's flexibility, its ability to simulate different memory mechanisms, and its potential for bridging the gap between computational models and neuroscientific understanding.

The study of language acquisition and processing through behavioral experiments, particularly in humans, infants, and animals, presents significant challenges. While these experiments offer valuable insights, they are often limited in their ability to evaluate cognitive processes, typically focusing on outputs and evaluating them if they are significantly different from random [185].

To address these limitations and provide a more comprehensive understanding of language learning mechanisms, we propose an extensible computational modelling framework. This framework allows for the systematic investigation of various neural network models, inductive biases, and memory mechanisms, enabling researchers to observe and analyze generalization results across different conditions.

6.8.1 Framework Advantages

The proposed framework offers several advantages:

- **Flexibility:** Flexibility in modifying model architectures, inductive biases, and memory mechanisms
- **Parameter investigation:** Ability to investigate the effects of various parameters on generalization outcomes, including memory constraints, data characteristics, task complexity, learning parameters, curriculum design, prior knowledge (through pre-training from models of previous tasks).
- **Comprehensive analysis** of simulation experiment results through generalization results and rates, statistical analysis of parameter effects, visualization and analysis of behavioral outputs allowing for the study of model errors, examination of memory access patterns, analysis of generation and acceptance/rejection patterns to identify learned and unlearned rules. It also provides internal representations that can be compared to imaging studies via methods such as regression [186] and representational similarity analysis [187].
- **Explainability:** With the help of comprehensive analysis tools mentioned above, it offers to understand neural networks or theorize about natural systems that would otherwise be considered a black box.
- **Falsifiability:** Framework uses a strict criteria for learning and generalization allowing it to falsify the model or framework behavior on conditions where errors occur. This paves the way for further improving the framework or re evaluating the assumptions or theories.

6.8.2 Flexibility and Comparative Analysis

One of the key strengths of this framework lies in its ability to implement and compare different neural network architectures, including Long Short-Term Memory (LSTM) networks, Legendre Memory Units (LMUs), and Neural Turing Machines (NTMs). This flexibility allows for a comprehensive exploration of how different architectures perform on various cognitive tasks, particularly in the domain of language processing and artificial grammar learning. Framework allows these architectures to be modified as neural network models and also supports the inclusion of different kinds of neural network mechanisms to model different aspects of computation and memory.

The framework facilitates direct comparisons between these architectures on identical tasks, providing valuable insights into their respective strengths and limitations. For instance, our experiments revealed that while LSTMs excel in managing long-term dependencies through their gating mechanisms, NTMs demonstrate superior performance in tasks requiring explicit external memory access. These comparisons not only highlight the unique capabilities of each architecture but also contribute to our understanding of the computational requirements for different cognitive processes.

6.8.3 Simulation of Cognitive Memory Mechanisms

A significant aspect of this framework is its capacity to simulate different types of memory processes, drawing parallels with cognitive and neuroscientific theories of memory. The distinct features of each architecture allow for the modeling of various memory mechanisms:

1. LSTMs, with their gating mechanisms, provide an effective model for working memory processes, particularly in tasks requiring the retention and manipulation of information over extended periods.
2. LMUs offer a unique approach to continuous-time representation, potentially aligning with theories of time-based memory decay and retrieval in cognitive psychology.
3. NTMs, with their external memory and content-based addressing, present an intriguing parallel to hippocampal memory processes, particularly in their ability to store and retrieve specific information based on content similarity.

The ability to simulate these diverse memory mechanisms within a unified framework provides a powerful tool for exploring hypotheses about memory functioning in cognitive tasks. For example, the success of NTMs in tasks requiring precise recall of earlier information mirrors the role of episodic memory in human cognition, suggesting potential parallels between the model's external memory and hippocampal function in the brain.

6.8.4 Cognitive Task Modeling

Our framework has demonstrated its efficacy in modeling a range of cognitive tasks, with a particular focus on artificial grammar learning. The performance of different architectures on these tasks offers insights into their suitability for modeling specific cognitive functions:

1. RNNs, serving as a baseline, showed proficiency in simple sequential tasks but struggled with longer dependencies.
2. LSTMs excelled in tasks requiring the retention of information over longer sequences, aligning with their known strengths in natural language processing tasks.
3. LMUs demonstrated efficiency in handling temporal dependencies, suggesting their potential utility in modeling time-sensitive cognitive processes.
4. NTMs showed remarkable capabilities in tasks requiring precise recall and manipulation of stored information, particularly excelling in complex grammar learning tasks that involve hierarchical structures.

These findings not only showcase the framework's utility in cognitive task modeling but also highlight how different architectural features align with specific cognitive processes. The ability of NTMs to generalize complex grammatical rules, for instance, provides a computational parallel to the human capacity for abstract rule learning in language acquisition.

6.8.5 Bridging Computational Models and Neuroscience

A key contribution of this framework is its potential to bridge the gap between computational models and neuroscientific understanding. By providing a platform for implementing biologically inspired mechanisms (such as the attention mechanisms in NTMs), the framework allows for the exploration of how artificial neural networks might approximate brain functions.

The comparison between NTM's memory access mechanisms and hippocampal processes is of interest. Content-based addressing in NTMs has some similarities with pattern completion processes observed in the hippocampus, while the sequential addressing might parallel the temporal coding capabilities of hippocampal neurons. These parallels, while not exact, suggest potential avenues for generating testable hypotheses about memory processes in the brain.

However, it's crucial to acknowledge the limitations in drawing direct comparisons between artificial models and biological systems. The simplifications inherent in our models mean that while they can provide valuable insights, they should not be considered exact replicas of brain processes.

6.8.6 Extensibility and Future Directions

The modular nature of this framework allows for future extensions, including the incorporation of new architectures or memory mechanisms as they are developed. This extensibility ensures that the framework can evolve alongside advancements in both machine learning and cognitive neuroscience.

Future research directions could include:

1. Combining various neural network models with external memory mechanisms
2. Incorporating more biologically detailed models to further bridge the gap with neuroscience.
3. Expanding the range of cognitive tasks to include more complex language or decision-making scenarios.
4. Developing methods to more directly compare model activations with neuroimaging data.
5. Developing ways to modify or introduce noise to certain mechanisms and see how disruption affects the results.

6.8.6.1 Towards More Biologically Plausible Systems

One of the steps in advancing neurocomputational models of cognitive functions can be to develop memory and control mechanisms that more closely resemble those found in biological systems. This approach aims to narrow the gap between artificial neural networks and the complex operations of the human brain. By incorporating more biologically realistic elements into our models, we can gain deeper insights into cognitive processes and create more accurate simulations of brain function. This direction not only improves the validity of our computational models but also allows for more meaningful comparisons between artificial and biological neural systems.

Spiking Neural Networks (SNNs)

Spiking Neural Networks (SNNs) offer a promising path towards more biologically plausible computational models. These networks aim to simulate neurons in a way that closely mimics the behavior of biological neurons. Unlike traditional artificial neural networks that process information in discrete time steps, SNNs operate continuously, better reflecting the ongoing activity in biological neural networks. This continuous processing allows SNNs to capture the temporal aspects of neural information processing more accurately.

An interesting possibility in this field is the conversion of existing artificial neural networks (ANNs) to SNNs. While this conversion doesn't fully explain how biological learning occurs, it's an important step towards creating mechanisms that more closely resemble actual brain tissue. This approach could reveal insights about brain function that traditional ANN models might miss.

SNNs also allow us to develop more realistic neuron-like mechanisms. These could include better models of synaptic plasticity, neural adaptation, and the complex interplay between excitation and inhibition in neural circuits. By incorporating these biologically inspired features, SNNs could offer a more nuanced understanding of cognitive processes and brain function.

As research in this area advances, integrating SNNs into cognitive models could lead to more accurate simulations of various cognitive functions, from perception and memory to decision-making and language processing. This increased biological realism may also help bridge the gap between computational models and neurophysiological data, potentially sparking new ideas and experiments in neuroscience.

Biologically Plausible Learning Methods Learning Methods

The biological plausibility of artificial neural networks (ANNs) has been a subject of debate, with one of the key arguments against it being the use of the backpropagation algorithm. While backpropagation is a powerful learning method, it is computationally expensive and requires weight transport, which may not be biologically realistic.

Fortunately, there are other learning methods for neural networks that are more biologically plausible. These methods include feedback alignment (FA) and its variants, target propagation, predictive coding, neuromodulated plasticity, contrastive Hebbian learning, and spike-timing-dependent plasticity (STDP) for spike networks. [188, 189]

Feedback alignment (FA) is a biologically plausible learning method that has been successfully applied to relatively large networks. It is computationally efficient and does not require weight transport [190]. Target propagation is another biologically plausible learning method that has shown promise for training deeper networks. It avoids the need for exact gradients and can work with discrete activation functions [188].

Predictive coding is a biologically plausible learning method that has traditionally been used in smaller networks. However, recent work has shown that predictive coding can be scaled to train larger networks on complex tasks like image classification [191]. Neuromodulated plasticity is a biologically plausible learning method that has potential for scaling, especially when combined with other local learning rules. It is particularly promising for reinforcement learning scenarios [192].

Contrastive Hebbian learning is a biologically plausible learning method that has been shown to work on moderately sized networks. It has theoretical connections to backpropagation, making it a promising alternative for biologically plausible learning [193]. Spike-timing-dependent plasticity (STDP) is a biologically plausible learning method that can be implemented for spike networks [194].

These biologically plausible learning methods offer promise for the development of more biologically realistic neural networks. As research in this area continues to improve, we may see the development of neural networks that are not only powerful but also more closely aligned with the workings of the human brain.

6.8.6.2 Improving Task Variety in Framework

At this stage, the framework is designed to handle artificial grammars and regulates complexity by enhancing grammatical intricacies. However, memory mechanisms are crucial not only for syntax but also for diverse types of tasks. This framework has the potential to be refined in future endeavors to handle various task definitions and scenarios.

Algorithmic tasks

A sequential learning approach may be used for tasks other than artificial grammar learning on different tasks using discrete representations. This might be used to study the effect of different memory mechanisms. These tasks can be any sequence processing task that its inputs and outputs can be represented as binary encodings. Some examples might include:

- **Path Finding:** finding a path or the shortest path in a given graph or maze. This might force the system to learn better spatial representations and remembering explicit places and nodes that have been visited. Complexity can be controlled by increasing the graph size.
- **Tree Traversal Algorithms:** Perform in-order, pre-order, or post-order traversals of binary trees. Complexity can be controlled by increasing tree depth and introducing unbalanced trees.
- **Parsing and Evaluation of Mathematical Expressions:** Parse and evaluate arithmetic expressions with parentheses. Complexity can be controlled by expression length and introduction of functions

Natural Language Tasks

In the current state of the framework the focus is solely on the grammar. Other important aspects of the language such as semantics and pragmatics (context) are not modelled. To be able to use the framework in a natural language, these different aspects of the language should be modeled too. But integration of these aspects Does not allow for the rigorous testing methodology that is employed.

Nonetheless, performance-based studies can be conducted by incorporating other natural language processing mechanisms, such as large language models, into the framework. This approach allows us to explore how different memory mechanisms might influence a broader spectrum of human language.

By utilizing this framework, we can conduct experiments to evaluate how integrating various memory mechanisms influences the performance of natural language processing systems. This approach could provide valuable insights into the memory requirements of language tasks. Alternatively, we can analyze memory usage and access mechanisms, aiming to create or understand representations using this architecture. However, pursuing this avenue would necessitate significantly larger networks, making it more challenging to comprehend the mechanisms involved during these processes.

Agent Integration and Simulation Environments

Like the artificial grammar learning tasks in this thesis, we can evaluate rule learning and generalization capabilities using various neural memory mechanisms. By incorporating these memory architectures into reinforcement learning agents, we can investigate the impact of different memory mechanisms on diverse tasks.

While human language interaction with an environment, even a simulated one, presents a complex challenge, integrating memory mechanisms that exhibit certain capabilities of symbolic architectures can be a compelling research topic. This is particularly true for tasks that would greatly benefit from generalizing the problem by learning and applying rules.

Some examples for this kind of research might be listed as:

- **Abstract Reasoning in Game Environments:** Learning the rules and the movement patterns of games like chess, Sokoban, block pushing puzzles from game interactions
- **Navigation and Spatial Reasoning Tasks:** A grid-world environment where the agent must learn and generalize rules about object placement or movement patterns. Different memory architectures can be utilized for generalizing spatial rules
- **Multi-task Learning Environments:** An environment where the agent needs to switch between different tasks, each with its own set of rules. This kind of research might focus on how learning distinct rules might affect task switching performance
- **Hierarchical Task Learning:** A task where the agent must learn hierarchical sequences to perform tasks. This kind of task may test the effect of memory mechanisms on hierarchical sequences other than artificial grammars.

6.8.6.3 Advanced Memory Mechanisms

Current state of the framework supports a few kinds of memory. It supports local memory, that is similar to cortical memory that is local activations in a region as the recurrent memory of RNNs. By using the NTM architecture, it supports semantic-like memory using content based addressing, it also supports working-memory like structure also with NTM's location based addressing. Current understanding of human memory suggests humans have different kinds of memory mechanisms. As future work it can be possible to implement architectures that have similar functionality and mechanisms to different memory types. We can drive a lot of inspiration from how already existing architectures handle memory. With this framework we are able to provide a differentiable mechanisms that we can train

the models using these frameworks to see how artificial cognitive systems might use these memory mechanisms

Here are some possible examples of these memory mechanisms:

Long-term memory: This is not directly a different memory mechanism, but currently, models do not store any information after training is complete, and the training only stores information by changing the weights of the model. A separate persistent memory mechanism can be used to experiment with how the information can be shared between different runs or tasks. This can be applied to any memory mechanism.

Memory consolidation: There can be a separate consolidation mechanism that decides which short-term memory elements will be written to the persistent storage.

Memory degradation: Some specific computational models can be implemented and tested for how the information in the memory is forgotten or degraded. This may help in creating or testing theories about memory mechanisms.

Episodic memory: A separate episodic memory module that stores complete "episodes" or experiences. This will allow the model to recall specific past experiences and use them for current tasks. This could be particularly useful for tasks requiring context-dependent learning or one-shot learning.

Declarative memory: An implementation for a memory retrieval mechanism that can read or write to a structured database for storing facts and events. Especially with integration with mechanisms that allow us to work with natural language or other structured languages, this can allow us to simulate declarative memory in a similar way to ACT-R [71] or SOAR [195], but also having the benefits of being trainable by data by the virtue of being differentiable.

Procedural memory: Currently, rule learning is performed during training and achieved by updating the weights in the controller. To better model procedural memory, we can design and implement a specific architecture to learn and store rules, and while making inferences, use those specific rules or representations to perform the tasks. This will allow for more specific models for rule learning and generalization and can be used to research how previously learned rules or representations can affect multi-task learning performance. Similar to declarative memory, this can allow us to simulate declarative memory in a similar way to ACT-R [71] or SOAR [195].

Chunking: A separate specific mechanism that allows the system to chunk specific items can allow us to see the effects of chunking strategy on various tasks. If the chunking mechanism can be switched on or off, this can simulate the scenarios where chunking is used or not.

Spatial memory: A memory mechanism can be designed specifically to represent spatial locations in a similar way to our understanding of place cells and spatial representations. This makes the model useful in tasks like navigation that require spatial knowledge.

Advanced memory control and attention mechanisms: As the number and complexity of memory types increase in the architecture, it will require special attention mechanisms to choose what kinds of information to include.

Memory disruption mechanism: There can be specific mechanisms to disrupt certain kinds of memory mechanisms to see how the results are affected by that. This can be used to test theories explaining

how certain aphasia or workloads may disrupt certain processes by comparing the results with the epistemological findings.

6.8.7 Limitations and Considerations

While the framework provides a powerful tool for exploring cognitive functions computationally, it's important to note its limitations. The artificial nature of the tasks and the simplifications inherent in neural network models mean that caution must be exercised in extrapolating findings to human cognition. Additionally, the framework currently focuses primarily on language-related tasks, and its applicability to other cognitive domains requires further investigation. The scope and the limitations will be discussed in Section 6.9.

This computational modeling and simulation framework offers a versatile approach to exploring cognitive functions through the lens of different neural network architectures. By allowing for the implementation and comparison of various memory mechanisms, it provides a unique platform for investigating the computational underpinnings of cognitive processes. While acknowledging its limitations, the framework's ability to bridge computational modeling with cognitive and neuroscientific theories might open up exciting possibilities for future research in cognitive science.

6.9 Scope and Limitations of the Computational Modeling and Simulation Framework

The Computational Modeling and Simulation Framework aims to provide explanations at both the computational and algorithmic levels of Marr's levels of analysis [196]. This section outlines the scope and limitations of this framework.

6.9.1 Scope

6.9.1.1 Computational Level

At the computational level, the framework seeks to generalize and learn rules, recognizing that certain mechanisms are capable of this while others are not. Various machine learning algorithms can be trained for the goal of learning artificial and natural grammars. For example, Large Language Models (LLMs) can serve as a tool to understand language at the computational level [31], providing insights into the nature of the problem.

However, it's crucial to note that the inner workings of the human brain and transformer architectures are fundamentally different. A notable example is that human language processing is sequential [197], whereas transformers utilize self-attention mechanisms applied to all tokens in parallel [54].

This framework aims to explain the nature of inductive learning and artificial grammar in a limited environment but unlike transformers it aims to provide further explanations as explained in the following sections.

6.9.1.2 Algorithmic Level

The Computational Modeling and Simulation Framework primarily aims to provide explanations at the algorithmic level like most computational cognitive frameworks do. Most computational cognitive models operate at this level. The framework attempts to model certain aspects of the underlying mechanism.

One of them is, how the representations and information is moved, stored or processed. With different architectures at hand it is possible to control the locality and transfer of representations from/to modules of specialized functionality. It can also model the absence or presence of some mechanisms and functionality. For example the augmented memory mechanisms such as the memory in NTM, can allow the storage and recalling of explicit representations with separate specialized modules whereas classical RNNs such as LSTMs keep the information locally and information storage is an integral part of the processing mechanism.

Unlike transformers architecture mentioned in the previous section, this cognitive modeling framework aims to simulate mechanisms similar to the human brain. A notable example is the memory access mechanism, which parallels human brain processes and Neural Turing Machines (NTMs), as explained in Section 2.5.3.1. The framework also aims to model both capabilities and limitations of the process, as detailed in Section 6.1.4.1.

6.9.1.3 Implementation Level

As a connectionist system, it might be possible to derive some implementation-level insights. However, the framework does not claim biological plausibility as is.

6.9.2 Limitations

The framework has several limitations:

- It does not intend to explain all cognitive/linguistic capabilities. Only syntactic and rule learning aspects of language are modeled, leaving semantics and thus natural language out of scope at this stage.
- It is not theoretical linguistics research and does not aim to provide formal analyses or proofs.
- While it has some high-level plausibility arguments, it does not directly claim to be neurally plausible.
- The framework does not fully address the problem of control. It uses artificial neural network architectures with discrete time steps, as explained in Section 6.5.4. While solutions exist for this problem, such as Adaptive Computation Time for neural networks [184], they are not directly addressed in this framework.
- The claims are made mostly from a computational point of view focusing on the underlying computation and its mechanisms

- Provided models are not detailed models of complex functionality aiming to simulate all aspects of the process. Instead drawn similarities are suggestive of general properties of the process to allow the creation or testing of more specific theories.

In this framework, neural networks are used as abstract descriptions. It aims to model what is learnable with certain structural biases found in neural systems and specific limitations or conditions. These abstract descriptions can help us understand what representations or mechanisms offer a more tractable problem space when learning from a finite set of examples. The similarities drawn between these architectures and actual mechanisms are not direct models of the actual process, but rather tools to understand the mechanisms and representations that allow for a viable solution to the learning problem.

CHAPTER 7

CONCLUSION

7.1 Research Questions and Methodology

This thesis set out to investigate the role of memory mechanisms in neural networks as neurocomputational models of cognitive functions, with a particular focus on syntactic processing and artificial grammar learning. Our research was guided by several key questions:

1. How do different memory mechanisms in neural networks affect computational abilities in tasks requiring complex cognitive functions, particularly in grammar learning and rule generalization?
2. What are the key structural biases in memory networks needed for solving problems traditionally challenging for connectionist approaches?
3. How can the use of deep learning tools in modeling memory networks serve as a link between higher-level cognitive theories and neuroscientific understanding?
4. What role do memory mechanisms in neural networks play in computational models when compared to human memory systems like episodic and working memory?

To address these questions, we developed a cognitive simulation framework that allows for the systematic evaluation of various processing and neural network mechanisms. This framework provided a platform to conduct comparative analyses of different memory mechanisms and their effects on rule learning performance, focusing on architectures such as Long Short-Term Memory networks (LSTMs), Legendre Memory Units (LMUs), and Neural Turing Machines (NTMs).

The use of artificial grammars in our study offered several key advantages:

- The ability to study syntax separately from semantics.
- A precise evaluation of rule learning performance.
- The capacity to generate infinite data.
- A formal definition of the problem space.
- Easier analysis and control of complexity.

These features allowed us to isolate and study syntactic processes with a level of precision that would be challenging with natural language data, while still providing insights into the computational requirements for processing complex grammatical structures.

Our methodology involved training these various neural network architectures on a range of artificial grammars, from regular grammars to context-free and mildly context-sensitive grammars. We employed a stringent generalization criterion, ensuring that successful models truly learned and applied the grammatical rules rather than merely performing well statistically.

Through this approach, we aimed to not only compare the performance of different neural architectures but also to draw parallels between these computational models and theories of human language processing and memory systems.

7.2 Key Findings and Insights

Our research yielded several significant findings:

1. We observed a marked difference in the performance of neural networks with different memory architectures when learning and generalizing artificial grammars.
2. Neural Turing Machines (NTMs) and other architectures with external addressable memory demonstrated superior performance in learning and generalizing complex grammatical structures compared to recurrent neural networks with local memory, such as LSTMs and LMUs.
3. The results suggest that connecting external memory structures to grammar processing mechanisms appears to be crucial for processing complex grammars. This contrasts with approaches relying solely on recurrent local memory.
4. While NTMs and similar architectures were able to learn some context-sensitive tasks, they were not successful in all cases. Interestingly, their limitation appears to align with the category of mildly context-sensitive grammars, which is considered to be the complexity level of human languages.

It's important to note that our generalization criteria were exceptionally stringent. In our framework, generalization signifies the ability to learn and apply grammatical rules accurately, rather than merely performing well statistically. Models that achieved generalization in our studies can be considered both complete and sound in their understanding of the grammatical rules.

7.3 Implications for Cognitive Science and AI

The alignment between the grammatical complexity that our models can learn and the grammatical complexity of human language is particularly intriguing. This suggests potential parallels between our computational models and human cognitive architecture.

While this thesis primarily operates at the computational and algorithmic levels of analysis, we have drawn some parallels between our models and human brain structures and processes based on neu-

roscientific studies. These connections, while speculative, provide a promising direction for future research that could bridge computational models with biological neural systems.

Our findings contribute to the ongoing dialogue between connectionist and symbolic approaches to cognition. The success of memory-augmented neural networks in learning complex grammatical structures suggests a potential reconciliation between these two paradigms.

7.4 Broader Implications

The broader implications of this research extend into various domains of artificial intelligence and natural language processing:

- **Applications in AI and NLP:** The insights gained from this study can be applied to develop more advanced AI systems, particularly those involved in natural language processing tasks. By incorporating external memory mechanisms, these systems could achieve better understanding and generation of complex linguistic structures.
- **Human-like AI Systems:** The findings suggest that integrating memory networks with external addressable memory could lead to AI systems that mimic human cognitive processes more closely. Such systems would not only perform tasks more efficiently but also exhibit more human-like learning and reasoning capabilities.

7.5 Limitations and Future Directions

Despite the insights gained, it's important to acknowledge the limitations of our study. The use of artificial grammars, while beneficial for isolating syntactic processes, may not capture all the complexities of natural language processing. Future work should focus on extending these findings to more naturalistic language tasks.

Moving forward, several avenues for future research emerge:

1. Further validation of these findings against human behavioral and neuroimaging data.
2. Exploration of more complex grammatical structures and their relation to human language.
3. Investigation of how these insights might be applied to improve natural language processing systems.
4. Development of more sophisticated neurocomputational models that incorporate the principles uncovered in this study.

7.6 Conclusion

This research contributes to our understanding of the computational principles underlying complex cognitive functions, particularly in the domain of language processing. By demonstrating the impor-

tance of external memory mechanisms in learning and generalizing complex grammatical structures, we have provided insights that may inform both the development of more sophisticated AI systems and our understanding of human cognitive architecture. As we continue to unravel the complexities of cognitive mechanisms of language, the bridge between artificial neural networks and biological systems grows stronger, promising exciting developments in both cognitive science and artificial intelligence.

REFERENCES

[1] R. C. Berwick, A. D. Friederici, N. Chomsky, and J. J. Bolhuis, "Evolution, brain, and the nature of language," *Trends Cogn. Sci.*, vol. 17, pp. 89–98, 2 2013.

[2] B. Lipkin, G. Tuckute, J. Affourtit, H. Small, Z. Mineroff, H. Kean, O. Jouravlev, L. Rakocevic, B. Pritchett, M. Siegelman, and others, "Probabilistic atlas for the language network based on precision fMRI data from> 800 individuals," *Scientific Data*, vol. 9, no. 1, p. 529, 2022.

[3] P. Hagoort, "Nodes and networks in the neural architecture for language: Broca's region and beyond," *Current opinion in Neurobiology*, vol. 28, pp. 136–141, 2014.

[4] N. Ding, L. Melloni, H. Zhang, X. Tian, and D. Poeppel, "Cortical tracking of hierarchical linguistic structures in connected speech," *Nature Neuroscience*, 2015.

[5] A. D. Friederici, "The cortical language circuit: from auditory perception to sentence comprehension," *Trends in cognitive sciences*, vol. 16, no. 5, pp. 262–268, 2012.

[6] P. Hagoort, "The neurobiology of language beyond single-word processing," *Science*, vol. 366, no. 6461, pp. 55–58, 2019.

[7] A. D. Friederici, J. Bahlmann, S. Heim, R. I. Schubotz, and A. Anwander, "The brain differentiates human and non-human grammars: Functional localization and structural connectivity," *Proceedings of the National Academy of Sciences*, vol. 103, pp. 2458–2463, 2 2006.

[8] D. Perani, M. C. Saccuman, P. Scifo, A. Anwander, D. Spada, C. Baldoli, A. Poloniato, G. Lohmann, and A. D. Friederici, "Neural language networks at birth," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 108, no. 38, pp. 16056–16061, 2011.

[9] A. D. Friederici, "Evolutionary neuroanatomical expansion of Broca's region serving a human-specific function," *Trends in Neurosciences*, 2023.

[10] J. K. Rilling, M. F. Glasser, T. M. Preuss, X. Ma, T. Zhao, X. Hu, and T. E. J. Behrens, "The evolution of the arcuate fasciculus revealed with comparative DTI," *Nat. Neurosci.*, vol. 11, no. 4, pp. 426–428, 2008.

[11] S. Frey, S. Mackey, and M. Petrides, "Cortico-cortical connections of areas 44 and 45B in the macaque monkey," *Brain and Language*, 2014.

[12] J. Ardila, "Meaning in Language. An Introduction to Semantics and Pragmatics," *Journal of Pragmatics*, 2011.

[13] N. Chomsky, "Three Models for the Description of Language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[14] P. Bourreau, L. Kallmeyer, and S. Salvati, "On IO-copying and mildly-context sensitive formalisms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[15] L. K. Schiffer and A. Maletti, "Strong equivalence of tag and ccg," *Transactions of the Association for Computational Linguistics*, 2021.

[16] M. Steedman and J. Baldridge, "Combinatory Categorial Grammar," in *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, 2011.

[17] D. Kemmerer, *Cognitive neuroscience of language*. Routledge, 2022.

[18] B. Lipkin and E. Fedorenko, "The Brain Basis of Language Processing: A Review of Neuroimaging Evidence," *Annual Review of Linguistics*, vol. 8, pp. 273–297, 2022.

[19] S. Malik-Moraleda, D. Ayyash, J. Gallée, J. Affourtit, M. Hoffmann, Z. Mineroff, O. Jouravlev, and E. Fedorenko, "An investigation across 45 languages and 12 language families reveals a universal language network," *Nature Neuroscience*, vol. 25, no. 8, pp. 1014–1019, 2022.

[20] R. M. Braga, T. D. Griffiths, and M. A. L. Ralph, "The Human Connectome: An Overview," *Cortex*, vol. 126, pp. 279–293, 2020.

[21] E. Fedorenko, P.-J. Hsieh, A. Nieto-Castañón, S. Whitfield-Gabrieli, and N. Kanwisher, "New method for fMRI investigations of language: defining ROIs functionally in individual subjects," *Journal of neurophysiology*, vol. 104, no. 2, pp. 1177–1194, 2010.

[22] C. Pallier, A.-D. Devauchelle, and S. Dehaene, "Cortical Representation of the Constituent Structure of Sentences," *Proceedings of the National Academy of Sciences*, vol. 108, no. 6, pp. 2522–2527, 2011.

[23] E. Fedorenko, M. K. Behr, and N. Kanwisher, "Functional specificity for high-level linguistic processing in the human brain," *Proceedings of the National Academy of Sciences*, vol. 108, no. 39, pp. 16428–16433, 2011.

[24] X. Chen, J. Affourtit, R. Ryskin, T. I. Regev, S. Norman-Haignere, O. Jouravlev, S. Malik-Moraleda, H. Kean, R. Varley, and E. Fedorenko, "The human language system, including its inferior frontal component in "Broca's area," does not support music perception," *Cerebral Cortex*, vol. 33, no. 12, pp. 7904–7929, 2023.

[25] M. D. Hauser, N. Chomsky, and W. T. Fitch, "The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?," *Science*, vol. 298, pp. 1569–1579, 11 2002.

[26] W. Tecumseh Fitch and A. D. Friederici, "Artificial grammar learning meets formal language theory: An overview," 2012.

[27] W. T. Fitch, "Toward a computational framework for cognitive biology: Unifying approaches from cognitive neuroscience and comparative cognition," *Physics of life reviews*, vol. 11, no. 3, pp. 329–364, 2014.

[28] C. J. Donahue, M. F. Glasser, T. M. Preuss, J. K. Rilling, and D. C. Van Essen, "Quantitative assessment of prefrontal cortex in humans relative to nonhuman primates," *Proceedings of the National Academy of Sciences*, vol. 115, no. 22, pp. E5183–E5192, 2018.

[29] N. M. Schenker, W. D. Hopkins, M. A. Spocter, A. R. Garrison, C. D. Stimpson, J. M. Erwin, P. R. Hof, and C. C. Sherwood, "Broca's area homologue in chimpanzees (Pan troglodytes): probabilistic mapping, asymmetry, and comparison to humans," *Cerebral Cortex*, vol. 20, no. 3, pp. 730–742, 2010.

[30] D. Z. Jin, "Generating variable birdsong syllable sequences with branching chain networks in avian premotor nucleus HVC," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2009.

[31] G. Tuckute, N. Kanwisher, and E. Fedorenko, "Annual Review of Neuroscience Language in Brains, Minds, and Machines," 2024.

[32] G. Delétang, A. Ruoss, J. Grau-Moya, T. Genewein, L. K. Wenliang, E. Catt, C. Cundy, M. Hutter, S. Legg, J. Veness, and P. A. Ortega, "Neural Networks and the Chomsky Hierarchy," 7 2022.

[33] S. Imani and L. Du, "MathPrompter: Mathematical Reasoning using Large Language Models," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023.

[34] I. Lei and Z. Deng, "Hint of Thought prompting: an explainable and zero-shot approach to reasoning tasks with LLMs," 5 2023.

[35] R. Sun, "Beyond associative memories: Logics and variables in connectionist models," *Information Sciences*, 1993.

[36] I. Aleksander, "The logic of connectionist systems," in *Neural Computing Architectures*, pp. 133–155, The MIT Press, 3 1989.

[37] S. C. Schwering and M. C. MacDonald, "Verbal Working Memory as Emergent from Language Comprehension and Production," *Frontiers in Human Neuroscience*, 2020.

[38] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 10 2014.

[39] Z. Quan, W. Zeng, X. Li, Y. Liu, Y. Yu, and W. Yang, "Recurrent Neural Networks with External Addressable Long-Term and Working Memory for Learning Long-Term Dependences," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 813–826, 3 2020.

[40] A. Mali, A. Ororbia, and C. L. Giles, "The Neural State Pushdown Automata," *IEEE Transactions on Artificial Intelligence*, vol. 1, pp. 193–205, 9 2019.

[41] A. Newell, "YOU CAN'T PLAY 20 QUESTIONS WITH NATURE AND WIN: PROJECTIVE COMMENTS ON THE PAPERS OF THIS SYMPOSIUM," in *Visual Information Processing*, pp. 283–308, Elsevier, 1973.

[42] S. Lewandowsky and S. Farrell, *Computational Modeling in Cognition: Principles and Practice*. 2455 Teller Road, Thousand Oaks California 91320 United States: SAGE Publications, Inc., 2011.

[43] P. S. Churchland and T. J. Sejnowski, *The Computational Brain*. The MIT Press, 6 1992.

[44] M. Milkowski, *Explaining the Computational Mind*. The MIT Press, 3 2013.

[45] W. T. Fitch, *The evolution of language*. Cambridge University Press, 2010.

[46] A. D. Friederici, N. Chomsky, R. C. Berwick, A. Moro, and J. J. Bolhuis, "Language, mind and brain," *Nature Human Behaviour*, vol. 1, no. 10, pp. 713–722, 2017.

[47] D. J. Povinelli and J. Vonk, "Chimpanzee minds: suspiciously human?," *Trends Cogn. Sci.*, vol. 7, no. 4, pp. 157–160, 2003.

[48] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach (4th Edition)*. 2021.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[50] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986.

[51] P. Smolensky, "Connectionist AI, symbolic AI, and the brain," *Artificial Intelligence Review*, vol. 1, pp. 95–109, 1987.

[52] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 6 2015.

[53] A. Saxe, S. Nelli, and C. Summerfield, "If deep learning is the answer, what is the question?," *Nat. Rev. Neurosci.*, vol. 22, no. 1, pp. 55–67, 2021.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 5999–6009, 6 2017.

[55] T. Shallice and R. P. Cooper, *The organisation of mind*. Oxford University Press, USA, 2011.

[56] N. Kriegeskorte and P. Douglas, "Cognitive computational neuroscience," *Nature Neuroscience*, vol. 21, pp. 1148 – 1160, 2018.

[57] W. H. Zuidema, R. French, R. G. Alhama, K. Ellis, T. O'Donnell, T. Sainburg, and T. Gentner, "Five Ways in Which Computational Modeling Can Help Advance Cognitive Science: Lessons From Artificial Grammar Learning," *Topics in Cognitive Science*, vol. 12, pp. 925 – 941, 2019.

[58] S. Lewandowsky and K. Oberauer, "Computational Modeling in Cognition and Cognitive Neuroscience," in *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience*, pp. 1–35, Wiley, 3 2018.

[59] M. Pitt, I. J. Myung, and S. Zhang, "Toward a method of selecting among computational models of cognition.," *Psychological review*, vol. 109 3, pp. 472–91, 2002.

[60] M. A. Just, P. Carpenter, and S. Varma, "Computational modeling of high-level cognition and brain function," *Human Brain Mapping*, vol. 8, 1999.

[61] T. Griffiths, F. Lieder, and N. D. Goodman, "Rational Use of Cognitive Resources: Levels of Analysis Between the Computational and the Algorithmic," *Topics in cognitive science*, vol. 7 2, pp. 217–29, 2015.

[62] S. Palminteri, V. Wyart, and E. Koechlin, "The Importance of Falsification in Computational Cognitive Modeling," *Trends in Cognitive Sciences*, vol. 21, pp. 425–433, 2017.

[63] J. L. McClelland, "The Place of Modeling in Cognitive Science," *Topics in Cognitive Science*, vol. 1, pp. 11–38, 1 2009.

[64] I. Van Rooij, "The Tractable Cognition Thesis," *Cognitive Science*, vol. 32, pp. 939–984, 9 2008.

[65] R. C. O'Reilly and Y. Munakata, *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. 2000.

[66] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, 2017.

[67] R. G. M. Morris, *Parallel Distributed Processing: Implications for Psychology and }Neurobiology. Oxford University Press, USA, 1989.*

[68] R. Sun, "Artificial Intelligence: Connectionist and Symbolic Approaches," in International Encyclopedia of the Social & Behavioral Sciences *(N. J. Smelser and P. B. Baltes, eds.), pp. 783–789, Oxford: Pergamon, 1 2001.*

[69] R. Sun, "A Two-Level Hybrid Architecture for Structuring Knowledge for Commonsense Reasoning," in Computational Architectures Integrating Neural And Symbolic Processes*, pp. 247–281, Boston, MA: Springer US, 1995.*

[70] S. Wermter and V. Weber, "Learning Fault-tolerant Speech Parsing with SCREEN," in Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, (Seattle, Washington), pp. 670–675, AAAI Press, 1994.*

[71] J. R. Anderson and C. J. Lebiere, The atomic components of thought*. Psychology Press, 1998.*

[72] R. Sun and T. Peterson, "Learning in reactive sequential decision tasks: the {CLARION} model," in Proceedings of International Conference on Neural Networks ({ICNN'96})*, vol. 2, pp. 1073–1078 vol.2, ieeexplore.ieee.org, 1996.*

[73] S. Wermter and R. Sun, "An Overview of Hybrid Neural Systems," in Lecture Notes in Computer Science*, pp. 1–13, 2000.*

[74] S. Wermter and R. Sun, An overview of hybrid neural systems*. Springer, 2000.*

[75] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," Nature*, vol. 538, pp. 471–476, 10 2016.*

[76] L. Doumas and J. Hummel, "Computational Models of Higher Cognition," 2012.

[77] I. Farkas, "Indispensability of Computational Modeling in Cognitive Science," The Journal of Cognitive Science*, vol. 13, pp. 401–429, 2012.*

[78] N. Chater, "Connectionism and classical computation," Behavioral and Brain Sciences*, vol. 13, pp. 493–494, 1990.*

[79] T. Tsuzuki, T. Kawahara, and T. Kusumi, "Connectionist modeling of higher-level cognitive processes," Shinrigaku kenkyu: The Japanese journal of psychology*, vol. 72 6, pp. 541–555, 2002.*

[80] S. Hanson and D. Burr, "What connectionist models learn: Learning and representation in connectionist networks," Behavioral and Brain Sciences*, vol. 13, pp. 471–489, 1990.*

[81] M. G. Dyer, "Connectionism Versus Symbolism in High-Level Cognition," pp. 382–416, 1991.

[82] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, J. Grewe Dominik }and Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, T. Kavukcuoglu Koray }and Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree }search," Nature*, vol. 529, pp. 484–489, 1 2016.*

[83] M. Hahn, "Theoretical Limitations of Self-Attention in Neural Sequence Models," Transactions of the Association for Computational Linguistics*, vol. 8, pp. 156–171, 2020.*

[84] K. Fukushima, "Training multi-layered neural network neocognitron," Neural Netw.*, vol. 40, pp. 18–31, 4 2013.*

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput.*, vol. 9, pp. 1735–1780, 11 1997.*

[86] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 9 2014.*

[87] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv preprint arXiv:1609.08144*, 9 2016.*

[88] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 12 2013.

[89] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," Nature*, vol. 550, pp. 354–359, 10 2017.*

[90] A. Kiyonaga and T. Egner, "Working memory as internal attention: toward an integrative }account of internal and external selection processes," Psychon. Bull. Rev.*, vol. 20, pp. 228–242, 4 2013.*

[91] N. Hornstein and N. Chomsky, "Knowledge of Language: Its Nature, Origin, and Use," Philos. Rev.*, vol. 97, no. 4, p. 567, 1988.*

[92] N. Chomsky, Aspects of the Theory of Syntax*. The MIT Press, 1964.*

[93] L. M. Herman, D. G. Richards, and J. P. Wolz, "Comprehension of sentences by bottlenosed dolphins," 1984.

[94] R. Malassis, S. Dehaene, and J. Fagot, "Baboons (Papio papio) Process a Context-Free but Not a Context-Sensitive Grammar," Scientific Reports*, vol. 10, 2020.*

[95] C. ten Cate and K. Okanoya, "Revisiting the syntactic abilities of non-human animals: natural vocalizations and artificial grammar learning," Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 367, pp. 1984–1994, 2012.

[96] L. A. Heimbauer, C. M. Conway, M. H. Christiansen, M. Beran, and M. Owren, "Visual artificial grammar learning by rhesus macaques (Macaca mulatta): exploring the role of grammar complexity and sequence length," Animal Cognition, vol. 21, pp. 267–284, 2018.

[97] B. Wilson, M. Spierings, A. Ravignani, J. L. Mueller, T. H. Mintz, F. Wijnen, A. van Der Kant, K. Smith, and A. Rey, "Non-adjacent dependency learning in humans and other animals," Topics in cognitive science, vol. 12, no. 3, pp. 843–858, 2020.

[98] M. Rohrmeier, Q. Fu, and Z. Dienes, "Implicit Learning of Recursive Context-Free Grammars," PLoS ONE, vol. 7, 2012.

[99] F. H. Poletiek, H. Fitz, and B. R. Bocanegra, "What baboons can (not) tell us about natural language grammars," Cognition, vol. 151, pp. 108–112, 2016.

[100] M. Mohri and M. Nederhof, "Regular Approximation of Context-Free Grammars through Transformation," pp. 153–163, 2001.

[101] X. Jiang, T. Long, W. Cao, J.-r. Li, S. Dehaene, and L. Wang, "Production of Supra-regular Spatial Sequences by Macaque Monkeys," Current Biology, vol. 28, pp. 1851–1859, 2018.

[102] W. H. Zuidema, "Context-freeness Revisited," Cognitive Science, vol. 35, 2013.

[103] A. Ravignani, G. Westphal-Fitch, U. Aust, M. M. Schlumpp, and W. T. Fitch, "More than one way to see it: Individual heuristics in avian visual computation," Cognition, 2015.

[104] R. C. Berwick and N. Chomsky, Why Only Us: Language and Evolution. 2016.

[105] R. M. Braga, L. M. DiNicola, H. C. Becker, and R. L. Buckner, "Situating the left-lateralized language network in the broader organization of multiple specialized large-scale distributed networks," Journal of neurophysiology, vol. 124, no. 5, pp. 1415–1448, 2020.

[106] K. Mahowald and E. Fedorenko, "Reliable individual-level neural markers of high-level language processing: A necessary precursor for relating neural variability to behavioral and genetic variability," Neuroimage, vol. 139, pp. 74–93, 2016.

[107] A. A. Ivanova, Z. Mineroff, V. Zimmerer, N. Kanwisher, R. Varley, and E. Fedorenko, "The language network is recruited but not required for nonverbal event semantics," Neurobiology of Language, vol. 2, no. 2, pp. 176–201, 2021.

[108] G. Hickok and D. Poeppel, "The cortical organization of speech processing," Nature reviews neuroscience, vol. 8, no. 5, pp. 393–402, 2007.

[109] H. Goodglass and E. Kaplan, "The assessment of aphasia and related disorders," 1983.

[110] C. Rogalsky, "Broca's area, sentence comprehension, and working memory: an fMRI study," Frontiers in Human Neuroscience, vol. 2, no. October, pp. 1–13, 2008.

[111] A. Baddeley, Working Memory, Thought, and Action. Oxford University Press, 3 2007.

[112] G. Hickok, "The neural organization of language: evidence from sign language aphasia," Trends in Cognitive Sciences*, vol. 2, pp. 129–136, 4 1998.*

[113] Y. Nakai, J.-W. Jeong, E. C. Brown, R. Rothermel, K. Kojima, T. Kambara, A. Shah, S. Mittal, S. Sood, and E. Asano, "Three- and four-dimensional mapping of speech and language in }patients with epilepsy," Brain*, vol. 140, pp. 1351–1370, 5 2017.*

[114] E. Zaccarella and A. Friederici, "The neurobiological nature of syntactic hierarchies," Neuroscience & Biobehavioral Reviews*, vol. 81, pp. 205–212, 10 2017.*

[115] D. Poeppel, "The neuroanatomic and neurophysiological infrastructure for speech and language," Current Opinion in Neurobiology*, vol. 28, pp. 142–149, 10 2014.*

[116] E. Fedorenko and S. L. Thompson-Schill, "Reworking the language network," 2014.

[117] A. D. Friederici, "White-matter pathways for speech and language processing," in Handbook of Clinical Neurology*, 2015.*

[118] L. R. Chai, M. G. Mattar, I. A. Blank, E. Fedorenko, and D. S. Bassett, "Functional Network Dynamics of the Language System," Cerebral Cortex*, 2016.*

[119] P. Hagoort and P. Indefrey, "The neurobiology of language beyond single words," 2014.

[120] R. Kunert, R. M. Willems, D. Casasanto, A. D. Patel, and P. Hagoort, "Music and language syntax interact in Broca's area: An fMRI study," PLoS ONE*, 2015.*

[121] M. Musso, A. Moro, V. Glauche, M. Rijntjes, J. Reichenbach, C. Büchel, and C. Weiller, "Broca's area and the language instinct," Nature Neuroscience*, vol. 6, pp. 774–781, 7 2003.*

[122] A. Baddeley, "Working memory and language: an overview.," Journal of communication disorders*, vol. 36 3, pp. 189–208, 2003.*

[123] D. J. Acheson and M. MacDonald, "Verbal working memory and language production: Common approaches to the serial ordering of verbal information.," Psychological bulletin*, vol. 135 1, pp. 50–68, 2009.*

[124] E. S. Isbilen and M. H. Christiansen, "Chunk-Based Memory Constraints on the Cultural Evolution of Language," Topics in cognitive science*, 2018.*

[125] M. Ullman, "The Declarative/Procedural Model of Lexicon and Grammar," Journal of Psycholinguistic Research*, vol. 30, pp. 37–69, 2001.*

[126] J. R. Anderson, How can the human mind occur in the physical universe? *Oxford University Press, USA, 2007.*

[127] M. C. Duff and S. Brown-Schmidt, "The hippocampus and the flexible use and processing of language," Frontiers in Human Neuroscience*, vol. 6, no. MARCH 2012, p. 69, 2012.*

[128] N. J. Fortin, K. L. Agster, and H. B. Eichenbaum, "Critical role of the hippocampus in memory for sequences of events," Nature Neuroscience*, vol. 5, no. 5, pp. 458–462, 2002.*

[129] L. Davachi and S. DuBrow, "How the hippocampus preserves order: The role of prediction and context," 2015.

[130] H. Eichenbaum, "Time (and space) in the hippocampus," 2017.

[131] V. Piai, K. L. Anderson, J. J. Lin, C. Dewar, J. Parvizi, N. F. Dronkers, and R. T. Knight, "Direct brain recordings reveal hippocampal rhythm underpinnings of language processing," Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, pp. 11366–11371, 10 2016.*

[132] B. Opitz and A. D. Friederici, "Interactions of the hippocampal system and the prefrontal cortex in learning language-like rules," NeuroImage*, vol. 19, no. 4, pp. 1730–1737, 2003.*

[133] K. Henke, B. Weber, S. Kneifel, H. G. Wieser, and A. Buck, "Human hippocampus associates information in memory," Proceedings of the National Academy of Sciences of the United States of America*, 1999.*

[134] V. van de Ven, L. Waldorp, and I. Christoffels, "Hippocampus plays a role in speech feedback processing.," NeuroImage*, p. 117319, 2020.*

[135] K. Friston and G. Buzsáki, "The functional anatomy of time: what and when in the brain," Trends in cognitive sciences*, vol. 20, no. 7, pp. 500–511, 2016.*

[136] E. A. Kensinger, M. T. Ullman, and S. Corkin, "Bilateral medial temporal lobe damage does not affect lexical or grammatical processing: Evidence from amnesic patient H.M.," Hippocampus*, 2001.*

[137] D. G. MacKay, R. Stewart, and D. M. Burke, "H.M. revisited: Relations between language comprehension, memory, and the hippocampal system," Journal of Cognitive Neuroscience*, 1998.*

[138] P. Hagoort, "MUC (memory, unification, control) and beyond," Frontiers in Psychology*, 2013.*

[139] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.*

[140] F. F. A. Gers and E. Schmidhuber, "LSTM recurrent networks learn simple context-free and context-sensitive languages," IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.*

[141] A. R. Voelker, I. Kajić, and C. Eliasmith, "Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks," in Advances in Neural Information Processing Systems *(H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, pp. 15544–15553, Curran Associates, Inc., 2019.*

[142] E. Jefferies, H. Thompson, P. Cornelissen, and J. Smallwood, "The neurocognitive basis of knowledge about object identity and events: dissociations reflect opposing effects of semantic coherence and control," Philosophical Transactions of the Royal Society B*, vol. 375, 2019.*

[143] J. Davey, H. Edwards, G. Hallam, T. Karapanagiotidis, C. Murphy, I. D. Caso, K. Krieger-Redwood, B. Bernhardt, J. Smallwood, and E. Jefferies, "Exploring the role of the posterior middle temporal gyrus in semantic cognition: Integration of anterior temporal lobe with executive processes," Neuroimage*, vol. 137, pp. 165 – 177, 2016.*

[144] C. Kentros, N. Agnihotri, S. Streater, R. Hawkins, and E. Kandel, "Increased Attention to Spatial Context Increases Both Place Field Stability and Spatial Memory," Neuron*, vol. 42, pp. 283–295, 2004.*

[145] P. Seliger, L. Tsimring, and M. Rabinovich, "Dynamics-based sequential memory: winnerless competition of patterns.," Physical Review E*, vol. 67, p. 11905, 2002.*

[146] N. Yamamoto and A. Shelton, "Path information effects in visual and proprioceptive spatial learning.," Acta psychologica*, vol. 125 3, pp. 346–60, 2007.*

[147] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, 3 1994.*

[148] A. S. Reber, "Implicit learning of artificial grammars," Journal of Verbal Learning and Verbal Behavior*, vol. 6, pp. 855–863, 12 1967.*

[149] A. S. Reber, "Transfer of syntactic structure in synthetic languages," Journal of Experimental Psychology*, vol. 81, pp. 115–119, 7 1969.*

[150] V. C. Zimmerer, P. E. Cowell, and R. A. Varley, "Individual behavior in learning of an artificial grammar," Memory and Cognition*, vol. 39, pp. 491–501, 12 2011.*

[151] R. Gomez, L. Gerken, and R. Schvaneveldt, "The basis of transfer in artificial grammar learning," Memory & Cognition*, vol. 28, no. 2, pp. 253–263, 2000.*

[152] R. Schmidt, "Implicit learning and the cognitive unconscious: Of artificial grammars and SLA," Implicit and explicit learning of languages*, vol. 22, pp. 165–209, 1994.*

[153] P. Dayan and L. F. Abbott, "Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems," 2001.

[154] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," Current opinion in neurobiology*, vol. 14, pp. 481–487, 8 2004.*

[155] X. J. Wang, "Decision making in recurrent neuronal circuits," Neuron*, vol. 60, pp. 215–234, 10 2008.*

[156] T. L. Griffiths, N. Chater, C. Kemp, A. Perfors, and J. B. Tenenbaum, "Probabilistic models of cognition: exploring representations and inductive biases," Trends in Cognitive Sciences*, vol. 14, pp. 357–364, 8 2010.*

[157] M. J. Traxler, M. Boudewyn, and J. Loudermilk, "What's Special About Human Language? The Contents of the "Narrow Language Faculty" Revisited," Language and Linguistics Compass*, vol. 6, pp. 611–621, 10 2012.*

[158] N. Kriegeskorte, "Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing," Annual review of vision science*, vol. 1, pp. 417–446, 11 2015.*

[159] A. K. Joshi, K. V. Shanker, and D. Weir, "The Convergence Of Mildly Context-Sensitive Grammar Formalisms," 1990.

[160] L. Kallmeyer, "On Mildly Context-Sensitive Non-Linear Rewriting," Research on Language and Computation*, vol. 8, no. 4, pp. 341–363, 2010.*

[161] S. M. Shieber, "Evidence Against the Context-Freeness of Natural Language," pp. 320–334, 1985.

[162] A. K. Joshi, "Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?," in Natural Language Parsing *(D. R. Dowty, L. Karttunen, and A. M. Zwicky, eds.), Studies in Natural Language Processing, pp. 206–250, Cambridge University Press, 5 1985.*

[163] Y. Joshi Aravind K.
}and Schabes, "Tree-Adjoining Grammars," in Handbook of Formal Languages: Volume 3 Beyond Words *(A. Rozenberg Grzegorz*
}and Salomaa, ed.), pp. 69–123, Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.

[164] R. D. Hoeksema Jack
}and Janda, "Implications of Process-Morphology for Categorial Grammar," in Categorial Grammars and Natural Language Structures *(E. Oehrle Richard T.*
}and Bach and W. Deirdre, eds.), pp. 199–247, Dordrecht: Springer Netherlands, 1988.

[165] H. Seki, T. Matsumura, M. Fujii, and T. Kasami, "On multiple context-free grammars," Theoretical Computer Science*, vol. 88, no. 2, pp. 191–229, 1991.*

[166] S. M. McCauley and M. H. Christiansen, "Reappraising lexical specificity in children's early syntactic combinations," in Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 36, 2014.*

[167] A. Clark and R. Yoshinaka, "Distributional learning of context-free and multiple context-free grammars," Topics in grammatical inference*, pp. 143–172, 2016.*

[168] R. Michalski, "A Theory and Methodology of Inductive Learning," Artif. Intell.*, vol. 20, pp. 111–161, 1983.*

[169] K. Chan, A. Wong, and D. Chiu, "Discovery of probabilistic rules for prediction," in [1989] Proceedings. The Fifth Conference on Artificial Intelligence Applications*, pp. 223–229, IEEE Comput. Soc. Press, 1989.*

[170] PyTorch, "PyTorch: An Imperative Style, High-Performance Deep Learning Library." https://arxiv.org/abs/1912.01703, 2019.

[171] Loudinthecloud, "Neural Turing Machines (NTM) - PyTorch Implementation." https://github.com/loudinthecloud/pytorch-ntm, 2023.

[172] hrshtv, "A PyTorch implementation of Legendre Memory Units (LMUs) and its FFT variant." https://github.com/hrshtv/pytorch-lmu, 2023.

[173] Scikit-Optimize Contributors, "scikit-optimize v0.9.0." https://github.com/scikit-optimize/scikit-optimize/releases/tag/v0.9.0, 2021.

[174] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms Supplementary Materials," Proc. NIPS*, 2012.*

[175] D. Genkin, N. Francez, and M. Kaminski, "Mildly context-sensitive languages via buffer augmented pregroup grammars," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.*

[176] E. P. Stabler, "Varieties of crossing dependencies: Structure dependence and mild context sensitivity," Cognitive Science*, vol. 28, pp. 699–720, 9 2004.*

[177] M. Hamdouche and A. Sala, "Dynamic Kolmogorov Approach to RNN Universality," SSRN Electronic Journal*, 2021.*

[178] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.*

[179] D. Bhandarkar and D. Clark, "Performance from architecture: comparing a RISC and a CISC with similar hardware organization," pp. 310–319, 1991.

[180] S. Furber, "The advantages of RISC architectures," Computer Standards & Interfaces*, vol. 8, pp. 29–35, 1988.*

[181] S. O. Aletan, "An overview of RISC architecture," pp. 11–20, 1992.

[182] A. Newell and H. A. Simon, "Computer science as empirical inquiry," in ACM Turing award lectures*, p. 1975, New York, NY, USA: ACM, 1 2007.*

[183] T. R. Besold and K.-U. Kühnberger, "Symbolic and Hybrid Models of Cognition," in The Cambridge Handbook of Computational Cognitive Sciences *(R. Sun, ed.), Cambridge Handbooks in Psychology, pp. 139–172, Cambridge University Press, 2023.*

[184] A. Graves, "Adaptive Computation Time for Recurrent Neural Networks," 3 2016.

[185] B. Wilson, K. Smith, and C. I. Petkov, "Mixed-complexity artificial grammar learning in humans and macaque monkeys: Evaluating learning strategies," European Journal of Neuroscience*, 2015.*

[186] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant, "Encoding and decoding in fMRI," NeuroImage*, vol. 56, pp. 400–410, 5 2011.*

[187] N. Kriegeskorte, "Representational similarity analysis – connecting the branches of systems neuroscience," Frontiers in Systems Neuroscience*, 2008.*

[188] A. G. Ororbia and A. Mali, "Biologically Motivated Algorithms for Propagating Local Target Representations," Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4651–4658, 7 2019.*

[189] N. Caporale and Y. Dan, "Spike timing-dependent plasticity: A Hebbian learning rule," 2008.

[190] M. Akrout, C. Wilson, P. C. Humphreys, T. Lillicrap, and D. Tweed, "Deep learning without weight transport," in Advances in Neural Information Processing Systems*, 2019.*

[191] O. J. Henaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. Eslami, and A. V. O. Eslami, "Data-Efficient image recognition with contrastive predictive coding," in 37th International Conference on Machine Learning, ICML 2020*, 2020.*

[192] A. R. Daram, D. Kudithipudi, and A. Yanguas-Gil, "Task-Based Neuromodulation Architecture for Lifelong Learning," in Proceedings - International Symposium on Quality Electronic Design, ISQED*, 2019.*

[193] S. Schmidgall, "Adaptive reinforcement learning through evolving self-modifying neural networks," in GECCO 2020 Companion - Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020.*

[194] Y. Hao, X. Huang, M. Dong, and B. Xu, "A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule," Neural Networks*, 2020.*

[195] J. E. Laird, The Soar Cognitive Architecture*. The MIT Press, 4 2012.*

[196] D. Marr, A Computational Investigation into the Human Representation and Processing of Visual Information *. W.H. Freeman and Company}, 2010, mit ed., 1982.*

[197] P. F. Dominey, M. Hoen, J.-M. Blanc, and T. Lelekov-Boissard, "Neurological basis of language and sequential cognition: Evidence from simulation, aphasia, and ERP studies," Brain and Language*, vol. 86, pp. 207–225, 8 2003.*

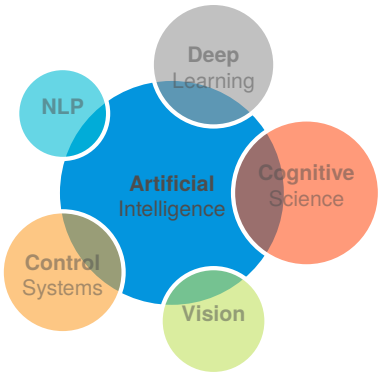
Sinan Onur ALTINUÇ

AI Specialist / Team Lead / Cognitive Scientist

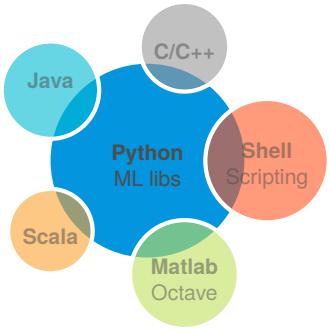
Fields of Interest

Artificial Int.(AI)
Machine Learning
Cognitive Architectures
Cognitive Science
Deep Learning
Data Science
Computer Vision
NLP
Synthetic Data Gen.

Technical Capabilities



Programming Languages



Experiences

03/23 - Now **Head of Perception and AI** — Buyutech
In charge of AI and perception teams of the company developing smart cameras. Web: `http://www.buyutech.com.tr`

05/18 - 02/23 **Director of Machine Learning and R&D** — OBSS
In charge of machine learning and R&D groups of the company, research topics ranging from NLP, computer vision, to autonomous vehicles, biomedical and environmental sensors processing. Providing solutions for various companies. Responsible for selecting, managing projects, team management along with research and development. Web: `http://www.obss.com.tr`

02/17 - 05/18 **Senior Software Engineer(Data Scientist)** — Unscrambl Software
In charge of adapting streaming analytics product to banking domain, building and integrating various analytics. Web: `http://www.unscrambl.com/`

09/14 - 08/16 **Research Assistant/Specialist** — Çankaya University
Assistant lecturer in computer engineering department, academic research.

07/10 - 09/14 **Software Engineer** — Environmental Tectonics Corporation (ETC)
Design and development of PC and contol software. Experience in working aboard (USA). Web: `http://www.etcusa.com/about/`

Projects

1. **Automative Vision System Projects**
   Designing implementation and integration of vision systems for automative cameras such ADAS and driver monitoring system cameras
2. **Various Research and Data Science Projects**
   Design and management of various customer specific forecasting, prediction, optimization and computer vision solutions for a variety of customers of OBSS including banks, logistics companies, and video platforms
3. **Drone Detection from Camera Images**
   Training and fine tuning various Deep Learning models on compiled datasets as well as complementary synthetic data. Using and tuning tracking algorithms to improve overall performance. We used a scoring algorithm to use the temporal information to increase accuracy. Got 1th place in AVSS 2021 Drone vs Bird Challenge. Building a drone detection product.
4. **Witwiser Online Proctoring Tool**
   Our team helped Witwiser to build computer vision capabilities for Witwiser that enables the system to detect suspicious conditions and objects. We also worked on edge computing solutions that allows machine learning models to run on users browser.
5. **Automated Large Area Surveillance**
   As a part of a computer vision competition organized by Turkish Presidency of Defence Industries we developed a computer vision system to effectively detect small objects on large images. We used a combination of provided real data and synthetic data generated from 3D simulation environment we created using game engines. Got 2nd place overall, 1st place for large vehicles category. We later we improved and open sourced the library on github: `https://github.com/obss/sahi`.

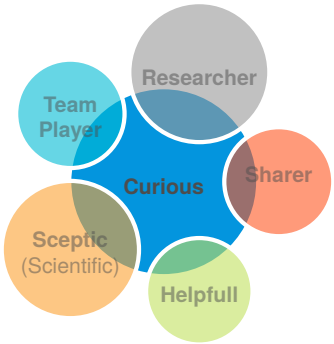
Languages

Turkish ★★★★★
English ★★★★½
German ★☆☆☆☆

OS preference

GNU/Linux ★★★★★
MacOS ★★★★☆
Windows ★★★☆☆

Personal Traits



Other Fields of Interesr

Music (Flute)
Photograpgy
Calligraphy
Education
Fantasy, Science Fiction
Role Playing Games
Popuar Science
Anime, Animation
Fountain Pens

6. **Fault Localization of Radar Sensory Data**
   Analysis and fault modelling of a radar system. Research and development of suitable features and distance metrics combining data analysis and expert knowledge.
7. **Question Generation from Natural Language Texts (Ongoing)**
   We built a system to automatically or semi automatically generate questions from given English or Turkish text. We use a collection of fine tuned transformer libraries and NLP libraries to flexibly generate meaningful questions in various domains.
8. **Autonomous Sea Vehicle (Ongoing)**
   Research, planing and preliminary development of autonomous sea vehicle. Planning and building simulation strategy, vision models for segmentation and object recognition, navigation, etc. Continuing as a part of EU project called ADRIATIC and internal research project ”Piribot”
9. **SIVA: Computer Vision Based Sports Assistant**
   Design and development of pose estimation based video analytics module for sports video platform, Sprongo `https://sprongo.com/`. We developed our analytics system as a scalable service capable of running multiple models at one server.
10. **Unscrambl Drive**
    Took part in development process of ’Drive’ which is Unscrambl’s solution for Marketing Automation using big data streaming and real time analytic. Was in charge of adaptation of the product and analytics for banking solutions.
11. **Wind Tower Sensor Analytics (Freelance Work)**
    Machine learning model to detect the icing and broken sensor conditions using various sensors of wind towers.
12. **Pattern Generalization using Neural Turing Machine (Ongoing)**
    Generalizing and testing different artificial grammar rules using Neural Turing Machines. Aim is to research which types of architectural biases can satisfy pattern generalization.
13. **Korean And US Air Force Pressure Chambers/(ETC)**
    Design and development of software of research altitude/pressure chambers manufactured for United States Air-force and Korean Air-force used for pilot training. Test and hardware-integration in USA office.

Education

2014 - 2024 **Cognitive Sciences PhD** — Middle East Technical University
GPA: 4.0/4, Thesis stage, doctoral proficiency exam successful.

2010 - 2013 **Computer Engineering MS** — Hacettepe University
GPA: 3.8/4 *Thesis :Semi-Automated Shoreline Extraction in Satellite Imagery and Usage of Fractals as Performance Evaluator, Hacettepe Üniversity, 2013*

2005 - 2010 **Computer Engineering** — Hacettepe University
GPA: 3.07/4

2001 - 2004 **High School** — Samsun Fen Lisesi (Science School)

Publications

- FC Akyon, E Akagündüz, SO Altinuc, A Temizel
  Sequence models for drone versus bird classification Sixteenth International Conference on Machine Vision (ICMV 2023), 13072, 56-61, 2024.

- D Cavusoglu, U Sert, S Sen, S Altinuc
  Jury: A Comprehensive Evaluation Toolkit arXiv preprint arXiv:2310.02040, 2023.
- FC Akyon, SO Altinuc, A Temizel
  Slicing aided hyper inference and fine-tuning for small object detection 2022 IEEE International Conference on Image Processing (ICIP), 966-970, 2022.
- F Cagatay Akyon, E Akagunduz, S Onur Altinuc, A Temizel
  Sequence Models for Drone vs Bird Classification arXiv e-prints, arXiv:2207.10409, 2022.
- O Eryuksel, KA Ozfuttu, FC Akyon, K Sahin, E Buyukborekci, D Cavusoglu, ...
  DroBoost: An Intelligent Score and Model Boosting Method for Drone Detection International Conference on Image Analysis and Processing, 399-409, 2022.
- A Coluccia, A Fascista, A Schumann, L Sommer, A Dimou, D Zarpalas, ...
  Drone-vs-bird detection challenge at ICIAP 2021 International Conference on Image Analysis and Processing, 410-421, 2022.
- FC Akyon, ALİDE ÇAVUŞOĞLU, C Cengiz, SO Altinuç, A Temizel
  Automated question generation and question answering from Turkish texts Turkish Journal of Electrical Engineering and Computer Sciences 30 (5), 1931-1944, 2022.

- A Coluccia, A Fascista, A Schumann, L Sommer, A Dimou, D Zarpalas, ...
  Drone-vs-bird detection challenge at IEEE AVSS2021 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 168, 2021.
- FC Akyon, O Eryuksel, KA Ozfuttu, SO Altinuc
  Track boosting and synthetic data aided drone detection 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 24, 2021.
- FC Akyon, D Cavusoglu, C Cengiz, SO Altinuc, A Temizel
  Automated question generation and question answering from Turkish texts arXiv:2111.06476
- Devrim Cavusoglu, Ogulcan Eryuksel, Sinan Altinuc
  Increasing Data Diversity with Iterative Sampling to Improve Performance Presented at Workshop on Data-Centric AI (NeurIPS 2021).
- FC Akyon, O Eryuksel, KA Ozfuttu, SO Altinuc
  Track Boosting and Synthetic Data Aided Drone Detection 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)
- A Coluccia, A Fascista, A Schumann, ..., Sinan Onur Altinuc
  Drone-vs-Bird Detection Challenge at IEEE AVSS2021 2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)
- S. Altınuç, A. Keçeli, E. Sezer
  "Semi-Automated Shoreline Extraction in Satellite Imagery and Usage of Fractals as Performance Evaluator" International Journal of Computer Theory and Engineering Vol. 6, No. 2, Nisan 2014
- S. Altınuç, A. Keçeli, E. Sezer
  "Semi-Automated Shoreline Extraction in Satellite Imagery and Usage of Fractals as Performance Evaluator" International Journal of Computer Theory and Engineering Vol. 6, No. 2, Nisan 2014

Organizations

BMO Chamber of Computer Engineers Education Commitie
HUBİT Hacettepe Unv. Computer Club (President, BoD) (2005-2010)

LKD Turkish Linux Users Association (Sponsor, University workgroups)
TBD-Genç (Turkey Informatics Association – Young) Hacettepe University Representative (2010)

References

Academic and private sector references are available upon request.

04 September 2024 / Sinan Onur ALTINUÇ