A REPRESENTATION-BASED
APPROACH TO CONNECT REGULAR
GRAMMAR AND DEEP LEARNING

A Dissertation in
Information Sciences and Technology
by
Kaixuan Zhang

©2021 Kaixuan Zhang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2021
The dissertation of Kaixuan Zhang was reviewed and approved by the following:

C. Lee Giles  
Professor of College of Information Sciences and Technology  
Dissertation Adviser  
Chair of Committee

Kenneth Huang  
Assistant Professor of College of Information Sciences and Technology

Shomir Wilson  
Assistant Professor of College of Information Sciences and Technology

Daniel Kifer  
Associate Professor of Department of Computer Science and Engineering

Mary Beth Rosson  
Professor of Information Sciences and Technology  
Director of Graduate Programs, College of Information Sciences and Technology
Abstract

Formal language theory has brought amazing breakthroughs in many traditional areas, including control systems, compiler design, and model verification, and continues promoting these research directions. As recent years have witnessed that deep learning research brings the long-buried power of neural networks to the surface and has brought amazing breakthroughs, it is crucial to revisit formal language theory from a new perspective. Specifically, investigation of the theoretical foundation, rather than a practical application of the connecting point obviously warrants attention. On the other hand, as the spread of deep neural networks (DNN) continues to reach multifarious branches of research, it has been found that the mystery of these powerful models is equally impressive as their capability in learning tasks. Recent work has demonstrated the vulnerability of DNN classifiers constructed for many different learning tasks, which opens the discussion of adversarial machine learning and explainable artificial intelligence. Therefore, it is imperative to apply formal language to facilitate the development of deep learning research in terms of these issues.

This dissertation focused on connections and interactions between formal language theory and deep learning research. First, we investigate fine-grained characteristics of regular grammar and deterministic finite automata (DFA) from a deep learning perspective. Then we aim to comprehend some of the mysteries of the vulnerability and explainability of DNN, design generic frameworks and deployable algorithms for verification. Following the logic, the dissertation contains the following three sections: regular grammar classification and learning with recurrent neural networks, topological analysis of sample influence and category-based analysis of grammar transfer, adversarial models for deterministic finite automata and verification of recurrent neural network.

In the first thread, we focus on how to differentiate regular grammar in terms of learning tasks. We introduce an entropy metric based on the concentric ring representation and categorized regular grammar into three disjoint subclasses. In addition, we provided classification theorems for different representations of regular grammar. Our second thread of study concentrates on the internal structure of regular grammar and applies a topological perspective to investigate the model-free sample influence. We develop a Shapley homology framework and propose two algorithms based on different Betti numbers. Furthermore, we established a category-based framework to probe into the mechanism of grammar transfer learning. In the third thread, we focus on the adversarial robustness of the recurrent neural
network (RNN). We generalize the adversarial sample framework to an adversarial model to study the fine-grained characteristics of DFA, including transition importance and critical patterns. Also, we propose a generic framework for verification and develop an algorithm under our framework and conduct a case study to evaluate the adversarial robustness of different RNNs on a set of regular grammars.

In summary, this research works as a bridge between regular grammar and machine learning to provide an open discussion on the topics and provide some guidance in practice, and we believe this is an auspicious beginning.
TABLE OF CONTENTS

List of Figures ............................................................... ix
List of Tables ................................................................. xi
Acknowledgements .......................................................... xii

1 Introduction ................................................................. 1
1.1 Research Objectives and Contributions ................................ 4
  1.1.1 Research Objectives ............................................... 4
  1.1.2 Contributions .................................................... 6
1.2 Road Map .................................................................... 7
1.3 Statement of Author Contribution ....................................... 8

2 Background ................................................................. 10
  2.1 Preliminaries ............................................................. 10
    2.1.1 Recurrent Neural Networks ...................................... 10
    2.1.2 Deterministic Finite Automata .................................. 11
    2.1.3 The Tomita Grammars ............................................. 12
  2.2 Related Work ............................................................ 12
    2.2.1 RNNs and Regular Grammars .................................... 13
    2.2.2 Rule Extraction for Recurrent Networks ...................... 14

3 Regular Grammar Classification and Learning with Recurrent Neural
   Networks ................................................................. 18
  3.1 Introduction .............................................................. 19
  3.2 Categorization of Regular Grammars .................................. 20
    3.2.1 Entropy of a Regular Language from an Concentric Ring Representation 20
    3.2.2 Set Representation ............................................... 24
4.6.2 Grammar Recognition .................................................. 69
4.7 Conclusion ........................................................................ 72
4.8 Appendix: Calculation of the Results for the Special Sets of Graphs ............................................ 73
  4.8.1 Results of Betti number $\beta_0$ ........................................ 73
  4.8.2 Results of Betti number $\beta_1$ ........................................ 74
5 Deep Learning, Grammar Transfer, and Transportation Theory ............................................. 76
  5.1 Introduction ................................................................. 77
  5.2 Preliminary ................................................................. 79
    5.2.1 Mathematical Preliminaries ........................................ 79
    5.2.2 Related Work ....................................................... 81
  5.3 Framework ................................................................. 82
    5.3.1 Grammar Transfer Learning ..................................... 82
    5.3.2 Category ............................................................. 83
    5.3.3 Framework .......................................................... 84
  5.4 Experiments ................................................................. 86
    5.4.1 Experiment Setup ................................................... 86
    5.4.2 Experiment Results ............................................... 87
  5.5 Discussion ...................................................................... 89
  5.6 Conclusion ...................................................................... 90
6 Adversarial Models for Deterministic Finite Automata and Verification of Recurrent Neural Network ............................................. 91
  6.1 Introduction ................................................................. 92
  6.2 The Formulation of the Adversarial Example Problem ......................................................... 94
  6.3 Transition Importance of a DFA ...................................... 95
    6.3.1 Transition Importance Estimation as an Adversarial Model Problem .......................... 96
    6.3.2 Transition Importance ............................................. 97
    6.3.3 Evaluation of DFA Transition Importance ......................................................... 100
  6.4 Critical Patterns of DFA .................................................. 101
    6.4.1 Different Types of Critical Patterns ................................... 101
    6.4.2 DFA Synchronizing Algorithm ................................... 104
    6.4.3 Evaluation of DFA Pattern Identification ......................................................... 106
6.5 Verification of Recurrent Neural Network . . . . . . . . . . . . . . . . . . . . 107
6.6 Evaluation of the Adversarial Robustness of Recurrent Networks . . . . . . . 109
6.7 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112

7 Conclusion and Future Work 113
7.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
7.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115

Bibliography 117
List of Figures

1.1 The road map of this dissertation. .............................................. 8

2.1 Examples of three Tomita DFAs. We depict the DFAs for grammars 2, 4, and 5. The red arrow indicates the initial state, and shaded circles indicate non-accept states. Dotted lines indicate input symbol “0” and solid lines indicate input symbol “1”. ......................................................... 13

2.2 Different Representations of Regular Grammar. .......................... 15

3.1 Flowchart of analysis. ............................................................... 20

3.2 Concentric ring representation of the distribution of strings of length $N$ ($1 \leq N \leq 8$) for seven Tomita grammars. Each concentric ring of a graph has $2^N$ strings arranged in lexicographic order, starting at $\theta = 0$. (We do not impose any constraint on the order of string arrangement; orders, e.g., gray codes, can also be selected for visualization.) ........................................... 21

3.3 Example for the Tomita-2 grammar where red (black) states are the accept (reject) states. ............................................................. 34

3.4 Error loss vs epochs on Tomita - 1, 3, 5 grammars. ...................... 36

3.5 Recurrent model hierarchy. ....................................................... 44

4.1 Framework of Shapley Homology for influence analysis ............... 51

4.2 An simple illustration of the Shapley Homology framework on grammar $1^* + 0^*(0 + 1)$. ................................................................. 57

4.3 The Čech complex, influence scores and entropy of the fourth grammar with $N = 4$ when applying Betti number $\beta_0$. ......................... 58

4.4 The Čech complex, influence scores and entropy of the fourth grammar with $N = 4$ when applying Betti number $\beta_1$. ......................... 60
4.5 Example graphs with classification results for each graph from left to right: 1, 2 and 1. This indicates the number of connected components recognized by a neural network on these graphs. 68

4.6 Accuracy of neural networks obtained on data sets with varying scales of manipulations. 68

4.7 Training Performance for Different Recurrent Networks on Grammars 1, 2 and 4. 70

5.1 Illustration of defined categories and functors. 85

5.2 Heatmaps for the grammar transfer learning performance and the estimated grammar transport learning. In both heatmaps, each entry represents the Wasserstein edit distance/learning distance/manipulation distance from the grammar indicated by its row index to the grammar indicated by its column index. 88

6.1 Example for the Tomita-7 grammar. Red (Black) states are the accept (reject) states. 95

6.2 Illustration of identified important transitions for example DFAs. The marked (with a yellow cross) and dashed lines demonstrate the most sensitive transitions of the original DFAs and the perturbed transitions, respectively. 101

6.3 An illustration of the difference between absolute and relative patterns. 102

6.4 DFA examples that illustrate Theorem 7 and 8. 105

6.5 Adversarial accuracy of an Elman-RNN for strings with varying length (from 100 to 200) on grammar 3. 111
## List of Tables

2.1 A description of several representative recurrent networks. ........................................ 11
2.2 Descriptions of the Tomita grammars. ................................................................. 12

3.1 Generalization to the Regular Grammars with an Arbitrary Alphabet Size. ...... 28
3.2 Analysis of Tomita Grammars. ................................................................. 33
3.3 Solutions for approximating DFA with 2-RNN, Elman-RNN, and MI-RNN. .... 42

4.1 Example grammars and their associated entropy values. ............................... 56
4.2 Six special sets of graphs and their values of influence for the vertices based
on Betti number $\beta_0$. .................................................................................. 57
4.3 Four special sets of graphs and their values of influence for the vertices based
on Betti number $\beta_1$. .................................................................................. 60
4.4 Parameter Settings For Neural Network Models and Datasets. .......................... 67
4.5 Entropy of different grammars with various radii ($K = 100$). ....................... 71
4.6 Accuracy of different RNNs obtained on data sets with noise. (The number
in bracket denotes the number of misclassified samples.) .................................... 72

6.1 Optimization results for the Tomita-3/5/7 grammars. ........................................ 101
6.2 Transition matrices for example DFAs. $A_\sigma(s_1, s_2)$ represents a transition from
state $s_1$ to $s_2$ via $\sigma$. We only provide the transitions matrices for DFA-3/4/5
and omit the matrices for DFA-1/2 (Tomita-4/7). .............................................. 106
6.3 Patterns and their corresponding confidence for example DFAs. When several
patterns have the same length, we randomly show only one of them. .............. 107
6.4 Verification results for positive and negative strings with the length of 200. .... 111
Acknowledgements

I would not have successfully completed this doctoral dissertation, as a milestone in my five years of study, without the support and guidance of various people at Pennsylvania State University. This dissertation is the result of many experiences I have encountered from dozens of remarkable individuals who I wish to acknowledge.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor C. Lee Giles, for his continuous support consistent support and guidance of my Ph.D. study. He continuously provided encouragement and was always willing and enthusiastic to assist in any way he could throughout the research project. His insights, encouragements, and guidance are one of the most valuable things I obtained during this journey. I consider it a privilege to have completed my doctoral degree under his guidance and to have learned from his research expertise.

Besides my advisor, I would like to thank the rest of my dissertation committee: Prof. Kenneth Huang, Prof. Shomir Wilson, and Prof. Daniel Kifer, for their encouragement, insightful comments, and hard questions.

I would like to express my gratitude to my collaborators, Professor Xinyu Xing, Professor Xue Liu, Dr. Qinglong Wang, and Mr. Wenbo Guo, etc. It is a great pleasure to work with them and take their insightful comments and suggestions for this dissertation and other research. I especially appreciate Dr. Qinglong Wang for his generous help on matters of numerous and varied points of assistance. I would like to thank my fellow doctoral students for their feedback, cooperation, and of course friendship. I have been very lucky to have such great friends and know amazing people in State College. I am grateful to all of them for making life more joyful and exciting for me.

Nobody has been more important to me in the pursuit of this degree than the members of my family. I am extremely grateful to my parents for their love, prayers, caring, and scaries for educating and preparing me for my future. In particular, my sincere appreciation goes to my beloved fiancée, Ye Wang, who has unceasingly and staunchly supported me for all these years, stood by me whenever my sadness shadows and my happiness appears, offered me the warmest embrace even from a thousand miles away. My family kept me going on and this dissertation would not have been possible without their input.
Chapter 1

Introduction

Computational complexity theory has developed rapidly in the past several decades, which are fundamental parts of current theoretical computer science [1]. They study the borders between possible and impossible in information processing, quantitative rules governing discrete computations has to be done and suffices to algorithmically solve various computing problems), algorithmic aspects of complexity, optimization, approximation, reducibility, simulation, communication, knowledge representation, information, etc [2]. The theory of automata and formal language, a branch of computational complexity theory, deal with designing abstract self-propelled computing devices that follow a predetermined sequence of operations automatically [3]. They provide a variety of concepts, methods, and tools to build the fundamentals of theoretical computer science. Furthermore, there have been many successful applications in control theory, compiler construction, parsing, and formal verification [4,5].

Formal languages theory has emerged from various origins [2]. It is quite obvious and natural that much of it has originated from linguistics. Specifically, the study of grammars initiated by Noam Chomsky [6] has opened new vistas in this development. Many parts of formal language theory have originated from modeling certain objects or phenomena. A model can be expressed by or identified with a language. Hence, we will not distinguish between automata theory and formal language theory and will use these two terms interchangeably. Another source is mathematics, in particular, certain problems in combinatorics, logic, and algebra of semigroups and monoids. A pioneer in this line of research was Axel Thue at the beginning of the 20th century; he investigated avoidable and unavoidable pat-
terns in long and infinite words [7]. Alan Turing [8] also proposed the general idea to find models of computing. A shining light of computer science research in the nineteen-sixties, a compulsory part of instruction in the discipline in the seventies and eighties, automata theory seems to have disappeared from lecture theatre and conference hall. Nonetheless, we find it, explicitly or implicitly, in the essence or the premises of a number of subjects in computer science that are currently new or fashionable [9]. This motivates us to revisit the formal language theory from a new perspective.

One of the promising perspectives is deep learning (DL). This specific meeting point enabled many researchers who were working on practical problems to induce or extract grammar from data [10]. Deep learning (DL) research undoubtedly plays one of the most critical roles in accelerating artificial intelligence (AI) research. It categorizes a collection of algorithms that utilize neural networks with deep architectures to effectively perform machine learning tasks that involve a massive amount of data at an unprecedented scale. The same amount of data usually is beyond the capabilities of classic machine learning models. Neural networks, with the capability of expressing arbitrary operation, ensures that the complex computational structures with multiple processing layers can represent almost any input data into corresponding multiple levels of abstraction with excellent performance [11]. The applications of deep learning have significantly improved the state-of-the-art in a variety of diverse research fields, computer vision, natural language understanding, including indoor localization, object detection, classification, and many other fields [11–14]. It seems that it can lead any domain to a new perspective level and has driven great enthusiasm among the academic world, not to mention the investment fever it brings.

An important reason for the success of DL research is that DNNs can automatically learn from enormous data their characteristics, representations, and interactions. However, the underlying mechanism remains unclear, which stimulated intensive research efforts in the field commonly termed by eXplainable Artificial Intelligence (XAI) [15]. Meanwhile, in recent years, there has been increasing research in a field known as adversarial machine learning [16] that concerns the robustness of learning models. One of the most studied problems is the adversarial example problem [17], which is also the problem discussed in this dissertation. This problem describes the phenomenon where a model, which generalizes well on clean datasets, is strikingly vulnerable to adversarial samples crafted by slightly perturbing clean samples.

Based on the discussion above, thinking about the role of formal language theory seems
to be a key issue under such a circumstance. Most research focused on some more practical problems of attempting to represent some symbolic knowledge by machine learning models [10], in contrast, limited research work tried to explore some profound results of formal language theory with machine learning techniques. Hence first, we will revisit formal language theory for fine-grained analysis. On the other hand, the theory of machine learning is far from complete, and it is observed that there have been plenty of mysteries of neural networks such as adversarial and explainable issues [18, 19]. We will then consider and discuss how formal language theory will facilitate us to establish a complete system and explore new possibilities. In this dissertation, we select a rather small portion of the automata theory to study. Namely, we will focus on deterministic finite automata and the corresponding regular grammar. Finite automata are the simplest model of machines, so simple that they take forms, appear in contexts, and hide in applications as numerous as they are diverse. To sum up, this dissertation mainly focused on the connections and interactions between regular grammar and deep learning. More specifically, Here we highlight the following key questions we attempt to answer in this dissertation:

1. What are the fine-grained characteristics of regular grammar and DFA in a deep learning perspective?

2. How to define a model-free sample influence and further investigate the mechanism of grammar learning?

3. How to utilize the DFA model for RNN verification?

The first question mainly considers the internal structure of regular grammar and the corresponding computational model DFA. Classifications and representations are two core issues in the traditional research field. In other words, we focus on classifying regular grammar based on different representations. And we also extend the adversarial sample to an adversarial model framework to investigate the transition importance and critical patterns of DFA. The second question adopted a topological perspective for a data manifold, i.e., decompose a topological feature to individual data points as their sample influence. Furthermore, based on the sample influence study, we proposed a category-based framework to understand the nature of transfer learning and towards revealing the mystery of machine intelligence. The third question mainly addresses the difficulties of adversarial samples problem and verification of sequential data, i.e., lack of appropriate oracles and proper distance metrics. The
dissertation can be seen as just a beginning to combine formal language theory and machine learning models, and it contains the following three sections: regular grammar classification and learning with recurrent neural networks (Chapter 3), topological analysis of sample influence, and category-based analysis of grammar transfer (Chapter 4, 5), adversarial models for deterministic finite automata and verification of recurrent neural network (Chapter 6). Note that we combine the fine-grained analysis of DFA with RNN verification since both of them are related to an adversarial sample scheme. We will have a brief discussion on the motivation, research objectives, and contributions respectively in the next section.

1.1 Research Objectives and Contributions

In this section, the research objectives of this dissertation will be presented, along with the achievements.

1.1.1 Research Objectives

Regular Grammar Classification and Learning with Recurrent Neural Networks

Recently there has been a resurgence of formal language theory in deep learning research. Most pragmatically oriented researchers worked on the more practical issues of attempting to extract or integrate symbolic knowledge in the system by machine learning, while limited research work focused on exploring the factors in grammar learning. On the other hand, although regular grammar can be significantly different, the family of regular grammar is usually studied as a whole for their computational power and compared with other formal languages in the Chomsky hierarchy. Hence the understanding of regular grammar remains at a relatively coarse-grained level. We believe it is still an open question regarding on internal structure of regular grammar. As such, in the first part of this dissertation, we aim at differentiating regular grammar in terms of learning tasks. Specifically, given enough positive and negative samples in a specific regular grammar, machine learning models can gradually identify its latent pattern and finally represent the regular grammar via the training process. The training process varies drastically and indeed reflects the difficulty of learning regular grammar, which highly depends on the complexity itself. Note that in order not to get embroiled in the diversity of factors in grammar learning, our investigation mainly focuses on the complexity of grammar.
Topological Analysis of Sample Influence and Category-based Analysis of Grammar Transfer

Many of the aforementioned methods study the sample influence problem through the lens of neural networks. Specifically, it is common to adopt a neural network (either a target model or a surrogate) to identify samples that are deemed as more influential by the neural network. While these research have yielded promising results, the scope of their offered model effectiveness is relatively limited. More specifically, the sample influence obtained with a surrogate model fails to sustain their effectiveness to other models. This implies that the results of several existing methods are highly dependent upon the given model. In this second line of work, we assume that the model is no longer provided. Thus, we aim at investigating the sample influence in a model-free rather than a model-agnostic manner. In addition, despite widespread adoption and success, artificial intelligence still suffers from some unexplainable phenomena. Machine intelligence indicates that learning models can complete the given task well and emphasize process information, which requires us to build the connection between machine intelligence and human intelligence. This motivates us to apply the sample influence to understand the foundations of transfer learning and reveal actual knowledge that is learned in the machine and deep learning systems.

Adversarial Models for Deterministic Finite Automata and Verification of Recurrent Neural Network

Most previous studies on estimating the adversarial robustness of neural networks target the models with feed-forward architectures and trained for image recognition tasks. Little has been done for recurrent networks. This is mainly due to the difficulty of formulating more rigorous constraints on the perturbation space required for sequential data. Moreover, it is challenging to introduce proper metrics for measuring the perturbation applied to sequential data. In this third line of work, we aim to address these challenges by proposing a framework that relies on rule extraction to evaluate the robustness of recurrent networks. On the other hand, we attempt to generalize the formulation of an adversarial sample to an adversarial model problem, then investigate the transition importance and critical patterns of DFA.
1.1.2 Contributions

In this dissertation, we have made the following contributions during the processes of achieving the previously mentioned research objectives. The publications related to these contributions are listed accordingly.

Contribution 1 In our first thread of work [20], we propose a new entropy metric for measuring the complexity of regular grammar in terms of machine learning which reflects the internal structure and hence provides a fine-grained understanding of regular grammar. We categorize regular grammars into three disjoint subclasses based on the entropy value with several classification theorems for different representations proved. Furthermore, we empirically demonstrate the validity of the proposed metric and corresponding classification by a series of experiments on the Tomita grammars. We provide a brief discussion on grammar learning with the connections between RNNs and DFAs from a theoretical perspective. We believe our investigation can facilitate a future rigorous understanding of grammar learning.

Contribution 2 In our second thread of work, we propose a generic framework to facilitate the development of sample influence as well as data complexity [21]. In our Shapley homology framework, we adopt a topological perspective to investigate the intrinsic properties of a data space. The key idea is to decompose the topological features of a complex and employ the Shapley value as the influence score to individual points. Under our framework, we develop two algorithms based on different Betti numbers that can be applied to scenarios where the underlying data space can be topologized by some specific metric. We theoretically and empirically validate the effectiveness of our algorithms. This opens the possibility that by calculating the importance of samples, we can acquire more general guidance for building various models that automatically capture the data distribution on their underlying manifold. Furthermore, we apply Shapley homology to build a category-based framework to investigate the mechanism of grammar transfer learning [22], with the intent of helping understand the foundation and intelligence of deep learning models. We believe this work will facilitate the research on machine learning model interpretation and explainable artificial intelligence.

Contribution 3 In our third thread of work, we propose a generic framework for using rules extracted from an RNN to estimate its adversarial robustness. We first propose to adopt a formal computational model – deterministic finite automata (DFA) – to represent
extracted rules. Then we have a thorough analysis of DFA by introducing a general scheme for adversarial models [23]. We mainly investigate the transition importance of a DFA through a model-level perturbation and critical patterns that can be used for identifying a specific DFA. Furthermore, we develop an algorithm for finding the critical patterns of a DFA by transforming this task as a DFA synchronizing problem and provide a theoretical approach for estimating the length of any existing perfect patterns. At last, we demonstrate using DFA to evaluate the adversarial robustness of different RNNs for regular grammars [24]. The experiments show the difference between the robustness of different RNNs and the difference in the difficulty of robust learning of regular grammars with different complexity. This work opens a discussion on the model-level analysis and verification of RNNs.

1.2 Road Map

The road map of this dissertation is presented in Figure 1.1. In Chapter 2, we first introduce some basic notions on RNNs and regular grammar, then provide a literature review on RNN, DFA and rule extraction. In Chapter 3, we present a theoretical analysis and empirical validation for subclasses of regular grammars to measure the complexity of regular grammar. We introduce an entropy metric and categorize regular grammar into three disjoint subclasses. In addition, we provided classification theorems for different representations of regular grammar.

In Chapter 4, we adopt a topological perspective to study the sample influence on data manifold. We propose the Shapley homology framework and design two algorithms based on different Betti numbers. We perform both analytical and empirical studies on several sets of data samples from two families of structured data. Furthermore, in Chapter 5, we follow the Shapley homology framework to present our work on investigating the mechanism of grammar transfer learning. We introduce our category-based framework, followed by detailed descriptions and evaluations.

In Chapter 6, we present a generic framework of using extracted DFA for evaluating the adversarial robustness of recurrent networks. We first extend the formulation, which is commonly adopted in existing work, of the adversarial example problem, to an adversarial model problem. With this scheme, we investigate the transition importance and critical patterns of DFA. Then we design an evaluation algorithm for RNN using extracted DFA, compare and demonstrate the adversarial robustness of different RNNs trained for learning
regular grammars.

In Chapter 7, we summarize this dissertation study and provide a discussion on future work.

1.3 Statement of Author Contribution

In the following, I provide a statement of the contributions made by co-authors in several published results used in this dissertation.

1. In the previous work [20], K.Z. and Q.W. conceived of the presented idea. K.Z. developed the analytical results and wrote the manuscript. Q.W. performed the evaluation on the proposed methods. C.G. supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

2. In the previous work [21], K.Z. developed the presented idea, verified the analytical results, and wrote the manuscript. Q.W. carried out the experiments and revised the
manuscript in the final submission. C.G. and X.L. supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

3. In the previous work [22], K.Z. devised the project, the main conceptual ideas, designed the experiments and wrote the manuscript. Q.W. performed the empirical evaluation. C.G. supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

4. In the previous work [23], K.Z. and C.G. proposed the presented idea. K.Z. developed the analytical results, performed the empirical evaluation, and wrote the manuscript. Q.W. helped in the discussion of the evaluation results. C.G. supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

5. In the previous work [24], Q.W. devised the project, designed and performed the experiments, and wrote the manuscript. K.Z. developed the theoretical formalism and helped in the experiment design. C.G. and X.L. supervised this project. All authors discussed the results and contributed to the final manuscript.
Chapter 2

Background

In general, this dissertation research mainly seeks to connect recurrent models to automata theory. In the following, we first provide a formal description of the recurrent neural network and regular grammars, along with necessary notations and detailed interpretations. Then we summarize existing complexity, representation of regular grammar, and rule extraction methods. With regard to the complexity of regular grammar, we will not only explain their details but also compare them with this dissertation research. Note that the lines of research applied several advanced mathematical tools as well, and we will review the related works in each chapter.

2.1 Preliminaries

In this section, we first introduce recurrent neural networks and several representative models. Then we provide a brief introduction of DFA and regular grammar (RG), followed by the Tomita grammars [25] used in this study.

2.1.1 Recurrent Neural Networks

Here we provide a unified view of the update activity of recurrent neurons for Elman-RNN [26], 2-RNN [27], MI-RNN [28], LSTM [29], and GRU [30]. These models have been frequently adopted either in previous work on DFA extraction or recent work on processing sequential data.

A recurrent network consists of a hidden layer $h$ containing $N_h$ recurrent neurons (an
individual neuron designated as $h_i$, and an input layer $I$ containing $N_I$ input neurons (each designated as $I_k$). We denote the values of its hidden layer at $t - 1$ th and $t$ th discrete times as $h^{t-1}$ and $h^t$. Then the hidden layer is updated by:

$$h^t = \phi(I^t, h^{t-1}, W),$$

where $\phi$ is the nonlinear activation function$^1$. $W$ denotes the weight parameters which modify the strength of interaction among input neurons, hidden neurons, output neurons, and any other auxiliary units. In most RNNs, $W$ usually comprise two separate weights, i.e. $U \in \mathbb{R}^{N_h \times N_I}$ and $V \in \mathbb{R}^{N_h \times N_o}$. The input $I^t$ and the hidden layer $h^{t-1}$ at the $t$ th and $t - 1$ discrete time are multiplied by weight $U$ and $V$, respectively. In Table 2.1, we present a detailed description of the hidden layer update for each RNN.

2.1.2 Deterministic Finite Automata

Based on the Chomsky hierarchy of phrase structured grammars [6], given an RG $G$, it is associated with one of the simplest automata, a DFA $r$, which can be described by a five-tuple $\{\Sigma, S, s_0, s_F, P\}$. $\Sigma$ is the input alphabet (a finite, non-empty set of symbols), and $S$ is a finite, non-empty set of states. $s_0 \in S$ represents the initial state, while $s_F \in S$ represents the set of final states. Note that $s_F$ can be the empty set, $\emptyset$. $P$ denotes a set of deterministic production rules. Every $G$ also recognizes and generates a corresponding regular language, which defines a set of strings of symbols from $\Sigma$. It is important to realize that DFAs cover a wide range of languages, which means that all languages whose string length and alphabet

$^1$Three typical activation functions include $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$, $\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\text{ReLU}(x) = \max(0, x)$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Hidden Layer Update</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elman-RNN [26]</td>
<td>$h^t = \phi(U I^t + V h^{t-1} + b)$</td>
<td>$U \in \mathbb{R}^{N_h \times N_I}, V \in \mathbb{R}^{N_h \times N_o}$</td>
</tr>
<tr>
<td>2-RNN [27]</td>
<td>$h^t = \phi(\sum_{i,j,k} W_{ijk} h_j^{t-1} I_k)$, $i,j \in 1, \ldots N_h, k = 1, \ldots N_I$</td>
<td>$W \in \mathbb{R}^{N_h \times N_h \times N_I}$</td>
</tr>
<tr>
<td>MI-RNN [28]</td>
<td>$h^t = \text{tanh}(\alpha \otimes U I^t \otimes V h^{t-1} + \beta_1 \otimes V h^{t-1} + \beta_2 \otimes U I^t + b)$</td>
<td>$U \in \mathbb{R}^{N_h \times N_I}, V \in \mathbb{R}^{N_h \times N_o}$, $\alpha \in \mathbb{R}^{N_h \times N_h}, \beta_1, \beta_2 \in \mathbb{R}^{N_h \times N_h}$</td>
</tr>
<tr>
<td>LSTM [29]</td>
<td>$s^t = \phi(U_s I^t + V_h h^{t-1})$, $s = {i, f, o, g}$ and $\phi = {\text{Sigmoid}, \text{Tanh}}$</td>
<td>$U_s \in \mathbb{R}^{N_h \times N_I}, V_h \in \mathbb{R}^{N_h \times N_h}$</td>
</tr>
<tr>
<td>GRU [30]</td>
<td>$z^t = \text{Sigmoid}(U_z I^t + V_h h^{t-1})$, $r^t = \text{Sigmoid}(U_r I^t + V_h h^{t-1})$, $g^t = \text{Tanh}(U_g I^t + V_h (h^{t-1} \odot r^t))$, $h^t = (1 - z^t) \odot g^t + z^t \odot h^{t-1}$</td>
<td>$U_z \in \mathbb{R}^{N_h \times N_I}, V_h \in \mathbb{R}^{N_h \times N_h}, s = {z, r, g}$</td>
</tr>
</tbody>
</table>
Table 2.2  Descriptions of the Tomita grammars.

<table>
<thead>
<tr>
<th>G</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1*</td>
</tr>
<tr>
<td>2</td>
<td>(10)*</td>
</tr>
<tr>
<td>3</td>
<td>an odd number of consecutive 1s are always followed by an even number of consecutive 0s</td>
</tr>
<tr>
<td>4</td>
<td>any string not containing “000” as a substring</td>
</tr>
<tr>
<td>5</td>
<td>even number of 0s and even number of 1s [35]</td>
</tr>
<tr>
<td>6</td>
<td>the difference between the number of 0s and the number of 1s is a multiple of 3</td>
</tr>
<tr>
<td>7</td>
<td>0<em>1</em>0<em>1</em></td>
</tr>
</tbody>
</table>

size are bounded can be recognized and generated by a DFA [27]. For a more thorough and detailed treatment of regular language and finite state machines, please refer to Hopcroft et al. [31]

2.1.3 The Tomita Grammars

The Tomita grammars [25] denote a set of seven RGs that have been widely adopted in the study of extracting DFA from RNNs. In principle, the Tomita grammars should be easily learnable, especially given that the DFAs associated with these grammars have between three and six states. Despite that the Tomita grammars are relatively simple, these grammars cover regular languages with a wide range of complexities. They have been widely studied and used as benchmarks [27, 32–34] in much work on learning grammars. These grammars all have alphabet \( \Sigma = \{0, 1\} \), and generate infinite languages over \( \{0, 1\}^* \). For each Tomita grammar, we refer to the binary strings generated by this grammar as its associated positive samples and the rest of binary strings as negative samples. In Table 2.2, we describe the positive samples accepted by all Tomita grammars. The visualization of DFAs for Tomita-2/4/5 is presented in Figure 2.1.

2.2 Related Work

In the remainder of this chapter, we shall endeavor to briefly discuss RNNs and regular grammars. We also revisit several complexity measure and some traditional representations of the regular grammar in automata theory. At last, we introduce existing rule extraction methods, especially the compositional approaches which are most widely adopted for extracting DFA
from recurrent networks.

2.2.1 RNNs and Regular Grammars

Grammars and automata and neural networks have been intertwined since McCulloch and Pitts’s early 1943 paper [36] which led to Kleene’s work [37] on regular grammars. Minsky’s dissertation [38] extended this to models of neural networks and automata. With the rebirth of neural networks, much work on recurrent networks and finite state automata [26,27,39,40] was restarted. Since then, there has been theoretical work on Turing equivalence [41] and finite state machine encoding and representation [42]. The computational hardness of some of the representation results has recently been discussed [43]. Recently, there has been a renewed interest in extracting learned DFA from recurrent neural networks [33,44,45]. Such methods can be important for verification for neural networks and explainable AI [24, 46].

There has been an increase in the natural language processing community on revisiting formal language models [47]. Because of their wide use and applications, we only focus on regular grammar.

Complexity of Regular Grammars

- Complexity of Shift Space. In symbolic dynamics [48], a particular form of entropy is defined to measure the “information capacity” of the shift space, which is a set of bi-infinite symbolic sequences that represent the evolution of a discrete system. When applied to measure the complexity of an RG, this entropy describes the cardinality of the strings defined by its language.
• Logical Complexity. RG can also be categorized according to logical complexity [49]: Strictly Local (SL), Strictly Piecewise (SP), Locally Testable (LT), etc. These classes have multiple characterizations in terms of logic, automata, regular expressions, and abstract algebra [50]. SL and SP languages are the simplest and most commonly used languages that define a finite set of factors and subsequences, respectively, and are selected to evaluate different RNNs on their performance in capturing the long-term dependency [50].

• Entropy and Regular Grammars. It is important to note that the concept of entropy has been introduced in the field of grammatical inference to solve different tasks [51,52] and more recently for distilling weighted automata [53]. Unlike the complexity of regular grammar, the thesis focuses on evaluating the complexity in terms of machine learning tasks, and our proposed definition originates in the graphical representation of regular grammar and gives insight into a metric for evaluating RNN learning and DFA extraction.

Representations of Regular Grammars  Automata theory has a close connection to many different mathematical subjects such as symbolic dynamics, group theory, algebraic geometry, topology, and mathematical logic [54]. Hence a regular grammar has different representations based on these perspectives. A description of several commonly used representations is shown in Figure 2.2. We will introduce the necessary concepts when they are used in the dissertation.

2.2.2 Rule Extraction for Recurrent Networks

According to a survey [55], most existing rule extraction methods can be categorized as: (1) compositional approaches, which categorize the cases when rules are constructed based on the hidden layers – ensembles of hidden neurons – of an RNN; (2) decompositional approaches, where rules are constructed based on individual neurons; (3) pedagogical approaches, which construct rules by regarding the target RNN as a black box and have no access to the inner state of this RNN; (4) eclectic approaches, which represent a hybrid of the above approaches. Most approaches mentioned above conduct rule extraction in a post hoc manner. In other words, rule extraction is performed with an already trained RNN and a data set containing samples to be processed by this RNN.
In several recent studies, we have seen the adoption of eclectic, decompositional, and compositional approaches. For example, a recent study [33] has proposed to build a DFA by both querying the outputs of an RNN for specific inputs and quantifying the continuous hidden space into a set of discrete hidden states. This method is effective for extracting DFAs with small size alphabets. However, similar to other methods, it cannot scale to DFAs with a large size alphabet. This is mainly due to the fact this method relies on the L* algorithm [56], which has polynomial complexity. As a result, the extraction process becomes extremely slow when a target RNN performs complicated analysis while processing sophisticated data [33]. Some other recent studies [44,57] have proposed to extract individual rules from data instances. Each extracted rule represents a pattern. Murdoch et al. [44,57] applied these approaches in the context of sentiment analysis. In this case, a pattern is a combination of important words identified by an RNN when processing a sentence. In order to construct a global rule set that describes the most critical patterns learned by an RNN, Murdoch et al. suggested aggregating extracted individual rules using statistical methods [44]. However, a rule set constructed in this manner usually lacks formal representation, hence may not be suitable for conducting a more thorough analysis of the behaviors of an RNN. In comparison with the eclectic and decompositional approaches mentioned above, the compositional approaches have been much more commonly adopted in previous studies [27,55,58,59]. More specifically, the following basic steps can describe a generic compositional approach:
1. Collect the values of an RNN’s hidden layers when processing every sequence at every time step. Then quantize the collected hidden values into different states.

2. Use the quantized states and the alphabet-labeled arcs that connect these states to construct a transition diagram.

3. Reduce the diagram to a minimal representation of state transitions.

These studies share a common hypothesis, which states that the state space of an RNN, which is well trained to learn an RG, should already be reasonably well separated with distinct regions that represent the corresponding states in some DFA [55]. This hypothesis, if true, implies that it would take much less effort to perform the quantization in the first step. Indeed, prior work has adopted various quantization approaches, including equipartition-based methods and different clustering methods, and demonstrated that this hypothesis holds for second-order RNN (2-RNN). Specifically, in an early study [27], an equipartition-based approach was proposed to cluster the hidden vector space of an RNN by quantizing the value of a hidden unit to a specified number of bins. For example, if we apply a binary quantization \(^2\) to the vector \(\{0.6, 0.4, 0.2\}\), we would obtain the encoding \(\{1, 0, 0\}\). One drawback of this form of quantization is that as the number of hidden units increases, the number of clusters grows exponentially. This computational complexity issue is alleviated if one uses clustering methods that are less sensitive to the dimensionality of data samples, e.g., \(k\)-means [58, 60, 61], hierarchical clustering [62], and self-organizing maps [63].

In the second step, in order to construct state transitions, prior work utilized either breadth-first search (BFS) approaches [27] or sampling-based approaches [63, 64]. The BFS approaches can construct a transition table that is relatively consistent but incurs high computation cost, especially when the size of the alphabet increases exponentially. Compared with the BFS approaches, the sampling approaches are computationally more efficient at the expense of introducing inconsistencies in the construction of a transition table. For a more detailed exposition of these two classes of methods, we refer the readers to this survey [55].

For the third step, prior studies commonly adopted a well-established minimization algorithm [31] for obtaining the minimal representation of a DFA. While much prior work has demonstrated the efficacy of compositional approaches when applied to SRN and 2-RNN,

\(^2\)Using a threshold value of 0.5, any value greater than 0.5 is assigned to the bin “1”, whereas other values less than or equal to this threshold are assigned to “0”.
however, little is known if these approaches can retain their efficacy for other RNNs, especially those that have demonstrated impressive performance on various sequence learning tasks, e.g., LSTM [29], GRU [30], and MI-RNN [28]. Another equally important yet missing study is how and if DFA extraction will be affected by the data source on which an RNN is trained.
Chapter 3

Regular Grammar Classification and Learning with Recurrent Neural Networks

Recently there has been a resurgence of formal language theory in deep learning research. However, most research focused on the more practical problems of representing symbolic knowledge by machine learning. In contrast, there has been limited research on exploring the fundamental connection between them. In order to obtain a better understanding of the internal structures of regular grammars and their corresponding complexity, we focus on categorizing regular grammars by using both theoretical analysis and empirical evidence in this chapter. Specifically, motivated by the concentric ring representation, we relaxed the original order information and introduced an entropy metric for describing the complexity of different regular grammars. We categorized regular grammars into three disjoint subclasses based on the entropy metric: the polynomial, exponential, and proportional classes. In addition, several classification theorems are provided for different representations of regular grammars. Also, our analysis is validated by examining the process of learning grammars with multiple recurrent neural networks. Our results show that as expected more complex grammars are generally more difficult to learn. We also provide some discussions on the comparison of different complexity measures and the model factors of grammar learning.
3.1 Introduction

Regular grammars (RGs) have been widely studied in theory of computation and intensively applied in natural language processing, compiler construction, software design, parsing, and formal verification [24, 65–67]. Despite their importance and pervasiveness, there is limited research [21, 50, 68] investigating the internal structure of regular grammars. As such, our understanding of regular grammar is relatively coarse-grained.

One approach to understanding a more fine-grained understanding of regular grammars is to investigate them through machine learning. Recent research has demonstrated that recurrent neural networks (RNNs) can achieve superior performance in a wide array of areas that involve sequential data [69], e.g., financial forecasting, language processing, program analysis, and particularly grammatical inference. Specially, recent work has shown that certain types of regular grammars can be more easily learned by recurrent networks [70, 71]. This is important in that it provides crucial insights in understanding regular grammar complexity. Furthermore, understanding the learning process of regular grammar also help differentiating different recurrent models [43, 72, 73].

This chapter will focus on establishing a closer connection between the complexity of regular grammar and machine learning tasks from both theoretical and empirical perspectives. From a theoretical perspective, we regard classification and representation as two fundamental problems. We follow previous work [75] by studying grammar from its concentric ring graph representation, which contains sets of strings (with a certain length) accepted and rejected by this grammar. Note that this representation can be used for any grammar. An entropy value is then introduced based on the properties of the concentric ring graph that categorizes all regular grammars into three classes with different levels of complexity and further establishes several classification theorems based on different representations of regular grammars. In addition, through an empirical study, different regular grammars are categorized by applying them to a set of learning tasks. That is, given enough positive (accepted) and negative (rejected) string samples of a specific regular grammar, it is expected that machine learning models will gradually identify a latent pattern of grammar through the training process. This shows that this indeed reflects the difficulty of learning regular grammar and highly depends on the complexity of grammar. All RNNs were evaluated and compared on string sets generated by different RGs with different levels of complexity so

1Izrail Gelfand once remarked, “all of mathematics is some kind of representation theory” [74].
as to explore the characteristics of each regular grammar. As such, the empirical results become well aligned with the theoretical analysis. We also provide a brief discussion on various definitions of entropy and the influence of recurrent models on grammar learning. It is hoped that these results can provide a deeper understanding of regular grammar under a learning scenario as well as practical insights into its complexity.

3.2 Categorization of Regular Grammars

This section reviews the concentric ring representation of regular grammar, and using this representation we introduce an entropy metric to evaluate their complexity. All RGs are then categorized into three classes according to their entropy values. Last, we provide several classification theorems of RGs in terms of their different representations. A flowchart of our analysis is shown Figure 3.1.

3.2.1 Entropy of a Regular Language from an Concentric Ring Representation

The concentric ring representation [75] of a regular grammar reflects the distribution of its associated positive and negative strings within a certain length. Specifically, in each concentric ring, all strings with the same length are arranged in lexicographic order where white and black areas represent accepted and rejected strings, respectively. We plot the concentric ring representations for Tomita grammars in Figure 3.2 to illustrate their differences. In each graph, every concentric ring contains the sets of strings and its following strings at a specific
Figure 3.2 Concentric ring representation of the distribution of strings of length $N$ ($1 \leq N \leq 8$) for seven Tomita grammars. Each concentric ring of a graph has $2^N$ strings arranged in lexicographic order, starting at $\theta = 0$. (We do not impose any constraint on the order of string arrangement; orders, e.g., gray codes, can also be selected for visualization.)

length that are accepted and rejected by its RG. Note that the percentages of accepted (or rejected) strings for different grammars are very different. For example, grammars 3 and 6 have the numbers of accepted strings much larger than that of grammar 1. This difference in prior empirical work [33, 45] showed that grammar 6 is much harder to learn than grammars 1 and 3. An intuitive explanation is that for grammar 6, flipping any 0 to 1 or vice versa means any accepted or rejected string can be converted into a string with the opposite label. An RNN needs to learn such subtle changes in order to correctly recognize all strings accepted by grammar 6. Since this change can happen to any digit, an RNN must account for all digits without neglecting any.

We now formally show that an RG that generates a more balanced set of accepted and rejected strings has a higher level of complexity and appears more difficult to learn. Given an alphabet $\Sigma = \{0, 1\}$, the collection of all $2^N$ strings of symbols from $\Sigma$ with length $N$ is denoted as $X^N$. For a grammar $G$, let $m_p^N (r_p^N)$ and $m_n^N (r_n^N)$ be the numbers (ratios) of positive and negative strings respectively. Relaxing the constraint that all strings are arranged in a lexicographic order, which indicates that all strings in $X^N$ are randomly distributed. We then denote the expected times of occurrence for an event $F_N$ – two consecutive strings having different labels – by $E[F_N]$. This gives the following definition of entropy for RGs.
with a binary alphabet.

**Definition 1 (Entropy).** Given a grammar $G$ with alphabet $\Sigma = \{0,1\}$, its entropy is:

$$H(G) = \limsup_{N \to \infty} H^N(G) = \limsup_{N \to \infty} \frac{1}{N} \log_2 E[F_N],$$

(3.1)

where $H^N(G)$ is the entropy calculated for strings with the length of $N$.

Furthermore, the following proposition to efficiently calculate the entropy by expressing $E[F_N]$ explicitly becomes:

**Proposition 1.**

$$H(G) = 1 + \limsup_{N \to \infty} \log_2 \left( r_p^N \cdot r_n^N \right) \frac{1}{N}. \quad \text{(3.2)}$$

**Proof of Proposition 1.** Given any concentric ring (corresponding to the set of strings with a length of $N$) shown in Figure 3.2, let $R$ denote the number of consecutive runs of strings and $R_p$ and $R_n$ denote the number of consecutive runs of positive strings and negative strings in this concentric ring respectively. Then we have $E[F] = E[R] - 1 = E[R_p] + E[R_n] - 1$. Without loss of generality, we can choose the first position as $\theta = 0$ in the concentric ring. Then we introduce an indicator function $I$ by $I_i = 1$ representing that a run of positive strings starts at the $i$-th position and $I_i = 0$ otherwise. Since $R_p = \sum_{i=1}^{2^N} I_i$, we have

$$E[R_p] = \sum_{i=1}^{2^N} E[I_i] \quad \text{and} \quad E[I_i] = \begin{cases} m_p/2^N, & i = 1 \\ m_n m_p/2^N (2^N - 1), & i \neq 1. \end{cases}$$

As such, we have

$$E[R_p] = \frac{m_p(1 + m_n)}{2^N} \quad \text{and} \quad E[R_n] = \frac{m_n(1 + m_p)}{2^N}.$$  

By substituting $E[F]$ into the entropy definition, we have

$$H(G) = 1 + \limsup_{N \to \infty} \frac{\log_2 \left( r_p^N \cdot r_n^N \right)}{N}. \quad \text{(3.3)}$$

\[\square\]

\[\text{Here we use } \limsup \text{ for certain particular cases, i.e when } N \text{ is set to an odd value for grammar 5.}\]
Thus the metric entropy is well-defined and lies between 0 and 1. Proposition 1 implies that the entropy of an RG is equal to the entropy of its complement, which confirms our intuition in terms of a learning task. Without loss of generality, we assume from now on that for all RGs, the set of accepted strings has a smaller cardinality. Also, we conclude that an RG generating more balanced string sets has a higher entropy value. As such, we can categorize all RGs with a binary alphabet based on their entropy values.

**Definition 2** (Subclass of Regular Grammar). Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have:

(a) $G$ belongs to **Polynomial** class if $H(G) = 0$;
(b) $G$ belongs to **Exponential** class if $H(G) \in (0, 1)$;
(c) $G$ belongs to **Proportional** class if $H(G) = 1$.

Formally speaking, the metric entropy defines an equivalence relation, denoted as $\sim_H$. A subclass of regular grammar can be considered as the equivalence class by the quotient $\text{RG}/\sim_H$, where $\text{RG}$ denotes the sets of all RGs. In this way, we have $\text{RG}/\sim_H = \{[\text{Po}], [\text{Ex}], [\text{Pr}]\}$, where $[\text{Po}], [\text{Ex}], [\text{Pr}]$ denote polynomial, exponential and proportional class respectively.

When compared to the entropy in shift space, which only considers accepted strings, our Definition 1 considers both the accepted and rejected strings. This is more informative and has other benefits. For example, given a data set with samples uniformly sampled from an unknown data set, we can then estimate the complexity of this unknown data set by calculating the entropy of the available data set. For a $k$-class classification task with strings of length $N$, let $m_i$ denote the number of strings in the $i$th class. Then we have $E[F_N] = 2^N - \frac{1}{2^N} \sum_{i=1}^{k} m_i^2$. We can then generalize the Definition 1 to a $k$-class classification case by substituting this in Definition 1. However, this can be challenging for the entropy defined for shift space since it can only be constructed in a one-versus-all manner. Also, the shift space cannot express all RGs since it deals with infinite sequences, and especially for grammars that lack shift-invariant and closure properties [48].

Additionally, $\limsup$ is crucial in the definition of entropy since there is no sufficient guarantee to the existence of the limit. It emphasizes describing the dynamical tendency when the length of blocks approaches infinity. For instance, the limit on grammar 5 cannot be obtained since the sequence becomes $0, 1, 0, 1, 0, 1, \cdots$ after $N$ exceeds certain number. This “periodic” property may also appear in some other grammars with restraints on length,
then only using lim fails to properly describe the complexity. Note that there must be an intrinsic period of a shift space of regular grammar when $N$ is sufficiently large, so another possible solution to this problem is the following definition:

**Definition 3.** Assume the minimum intrinsic period of a given shift space $X$ is $d$, then the generalized entropy of $X$ is defined as

$$H(X) = \frac{1}{d} \lim_{m \to \infty} \frac{1}{m} \log \sum_{i=md}^{(m+1)d-1} |B_i(X)|,$$

where $m \in \mathbb{Z}$ and $|B_i(X)|$ denotes the number of $i$-blocks appearing in points of $X$.

When $d = 1$, there is no difference between lim and lim sup. This Definition 3 can be interpreted as the summation over the additive group $\mathbb{Z}/m\mathbb{Z}$. Then the limit operation can be applied to it. In this way, applying lim sup in this scenario does not suffer the “periodic” problem mentioned above, since one can always find a convergent infinite subsequence from the original periodic sequence calculate the entropy.

### 3.2.2 Set Representation

Here we conduct a combinatorial analysis. Given a regular grammar $G$ we consider it as a set and explore the property of its cardinality. Namely we have the following theorem:

**Theorem 1.** Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have

(a) $G$ belongs to $[\text{Po}]$ if and only if $m_p^N \sim P(N)$, where $P(N)$ denotes the polynomial function of $N$;

(b) $G$ belongs to $[\text{Ex}]$ if and only if $m_p^N \sim \beta \cdot b^N$ where $b < 2$ and $\beta > 0$ and $H(G) = \log_2 b$;

(c) $G$ belongs to $[\text{Pr}]$ if and only if $m_p^N \sim \alpha \cdot 2^N$, where $\alpha \in [0, 1)$.

Here $\sim$ indicates that some negligible terms are omitted when $N$ approaches infinity.

**Proof of Theorem 1.** In both the Definition 1 and Proposition 1, lim sup is used to cover certain particular cases, for instance when $N$ is set to be odd value for grammar 5. In the following proof, without loss of generality, lim is used instead of lim sup for simplicity. According to Proposition 1, for any regular grammar $G$, its entropy $H(G) \in [0, 1]$. It can be checked that the maximum value of $H(G)$ is 1 when $r_p^N = 0.5$. Also, the minimum value of $H(G)$ is 0 and can be reached when $r_p^N = 0$ or 1. However, $r_p^N = 0$ or 1 are only allowed
for grammars that either accept or reject any string, hence are not considered this theorem. As such, in this case, the value of entropy is taken as the minimum when $r_p^N = 1/2^N$ or $1 - 1/2^N$. In the following, we only discuss the former case, and the latter can be similarly derived.

For each class of grammars, given that their $m_p$ takes the corresponding form shown in Theorem 1, the proof for the sufficient condition is trivial and can be checked by applying *L'Hospital’s Rule*. As such, in the following, we only provide proof for the necessary condition.

From (3.2), we have:

$$\begin{align*}
H(G) &= \lim_{N \to \infty} \log_2 \left( \frac{m_p \cdot 2^N - m_p^2}{N} \right) - 1 \\
&= \lim_{N \to \infty} \frac{m_p' \cdot 2^N + \ln 2 \cdot 2^N \cdot m_p - 2m_p \cdot m_p'}{\ln 2 \cdot (m_p \cdot 2^N - m_p^2)} - 1 \\
&= \lim_{N \to \infty} \frac{m_p' \cdot 2^N + \ln 2 \cdot m_p^2 - 2m_p \cdot m_p'}{\ln 2 \cdot m_p \cdot (2^N - m_p)},
\end{align*}$$

where $m_p'$ denotes the derivative of $m_p$ with respect to $N$. It is easy to check that $\lim_{N \to \infty} \frac{m_p'}{m_p}$ exists for regular grammars, then we separate the above equation as follows:

$$H(G) = \lim_{N \to \infty} \frac{m_p'}{\ln 2 \cdot m_p} + \lim_{N \to \infty} \frac{1 - \frac{m_p'}{m_p}}{\ln 2 \cdot m_p - 1}.$$ 

It should be noted that the second term in the above equation equals 0. Specifically, assuming that $m_p$ has the form of $\alpha \cdot b^N$ where $b < 2$ (b cannot be larger than 2 for binary alphabet), then the denominator of the second term is infinity. If $m_p$ has the form of $\alpha \cdot 2^N$, then the numerator tends to zero while the denominator is finite. As such, we have

$$H(G) = \lim_{N \to \infty} \frac{m_p'}{\ln 2 \cdot m_p}.$$ 

If $H(G) = 0$, then we have $\lim_{N \to \infty} \frac{m_p'}{m_p} = 0$, indicating that the dominant part of $m_p$ has a polynomial form of $N$ hence $m_p \sim P(N)$, where $P(N)$ denotes the polynomial function of $N$.

If $H(G) = t \neq 0$, then we have $\lim_{N \to \infty} \frac{\ln(m_p)}{2N \ln 2} = 1$, which gives that $m_p \sim \beta \cdot 2^{tN}$, where
\[ \beta > 0. \text{ If } t = \log_2 b, \text{ then we have } m_p \sim \beta \cdot b^N \text{ where } b < 2. \text{ Furthermore, if } t = 1, \text{ we have } m_p \sim \alpha \cdot 2^N \text{ where } \alpha \in [0, 1). \]

Theorem 1 shows the reason for naming these subclasses. In addition, when a specific order is posed on the set of given RG, we have a similar result from the number theory. Specifically, given a string \( s = x_1x_2x_3 \cdots x_N \) with length \( N \), we associate the string with a decimal number, i.e. \( D_s = \sum_{i=1}^{N} x_i \cdot 2^{i-1} + 2^N - 1 \). Note that this formula is different from the traditional approach to transforming a binary number to a decimal number since 0 has physical meaning in regular grammar. For example, 000 is different from 00 in regular grammar. That is the reason that we need an additional term \( 2^N - 1 \) to differentiate these situations. In this way \( D_s \) has induced an order on the regular grammar. Let \( D^n_s \) denote the \( n \)-th smallest number in this set, then we have the following result:

**Corollary 1.** Given any regular grammar \( G \) with \( \Sigma = \{0, 1\} \), we have
\begin{enumerate}
    \item (a) \( G \) belongs to [\( \text{Po} \)] if and only if \( \lim_{n \to \infty} B_n = 0 \);
    \item (b) \( G \) belongs to [\( \text{Ex} \)] if and only if \( \lim_{n \to \infty} B_n \in (0, 1) \);
    \item (c) \( G \) belongs to [\( \text{Pr} \)] if and only if \( \lim_{n \to \infty} B_n = 1 \).
\end{enumerate}
Here \( B_n = \log_2 n / \log_2 D^n_s \).

This corollary can be easily verified by the expressions of regular subclasses in Theorem 1 and the transformation introduced above. Here we provide a brief explanation of the Corollary 1. The denominator approximates the length of the string, and the numerator represents the cardinality of the set. Hence comparing with the original definition 1, this corollary provides an alternative perceptive of entropy. Take Tomita grammar 1 for example. The set is \( G = \{ \epsilon, 1, 11, 111, \cdots \} \) with the number \( D_s = \{0, 2, 6, 14, \cdots \} \), and the formula for \( D^n_s \) is given by \( D^n_s = 2^n - 2 \), hence by simple calculation, we find that the limit of the ratio is 0 when \( n \) approaches to infinity, which implies that grammar 1 belongs to the polynomial class. Note that in general, it is difficult to calculate the explicit formula for \( D^n_s \) and therefore, this corollary 1 has certain limitations in practical applications.

### 3.2.3 DFA Representation

Here we provide an alternative way to obtain the classifications for RGs using the transition matrices of its associated minimal DFA [76] in terms of states. This approach provides immediate results if minimal DFA is available. As such, this reduces the computation cost of
a data-driven approach. Here we again use the case when the alphabet size is two. However, it is easy to extend this for grammars with larger alphabets. Given a regular grammar $G$ with the alphabet $\Sigma = \{0, 1\}$ and its associated minimal complete DFA with $n$ states, let $T_0, T_1 \in \mathbb{Z}^{n \times n}$ denote the transition matrices associated with input 0 and 1, and the transition matrix of the DFA is defined as $T = T_0 + T_1$. Alternatively, the transition matrix of a given DFA can be considered as the transition matrix of an underlying directed graph by neglecting the labels associated with the edges. With the above settings, we have the following theorem:

**Theorem 2.** Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have

(a) $G$ belongs to $[\text{Po}]$ if and only if $k(T) = 1$ and $\sigma(T) = \{1, 2\}$;

(b) $G$ belongs to $[\text{Ex}]$ if and only if $k(T) = 1$ and $\sigma(T) - \{1, 2\} \neq \emptyset$, and $H(G) = \log_2 |\lambda_2|$ where $|\lambda_2|$ denotes the second largest modulus of the eigenvalues of $T$;

(c) $G$ belongs to $[\text{Pr}]$ if and only if $k(T) = 0$ or $k(T) = 2$.

Here $k(T)$ represents the number of diagonal elements equal to $2$ and $\sigma(T)$ denotes the set of modulus of all eigenvalues of $T$.

**Proof of Theorem 2.** We first introduce a lemma which is used in the proof.

**Lemma 1.** Let $v_1$ and $v_2$ denote two one-hot encoded column vectors corresponding to a starting state and an ending state of a DFA, respectively. One can construct an adjacent matrix $T$ for this DFA by regarding it as an undirected graph with each node represents a state and every edge represents the existence of a transition between a pair of states. Then the number of $L$-length strings that reach the ending state $v_2$ from $v_1$ is $v_1^T T^L v_2$.

Based on the lemma above, it is easy to see that the number of positive strings is $m_p = v_1^T T^L q$ where $q$ is an one-hot encoded column vector representing an accepting state. By applying Jordan decomposition to $P$, i.e., $T = SJS^{-1}$, we can see that $m_p = v_1^T ST^L S^{-1} q$ and $T^L$ is the only term depending on $L$. Specifically, take one Jordan block $J_i \in \mathbb{R}^{m \times m}$ with the eigenvalue of $\lambda$ and let $K_i$ denote the nilpotent matrix associated with $J_i$, that is, the superdiagonal of $K_i$ contains ones and all other entries are zero. Then we have:

$$J_i(\lambda)^L = (\lambda I + K_i)^L = \sum_{n=0}^{\min(L,m-1)} \binom{L}{n} \lambda^{L-n} K_i^n.$$ 

It is easy to see that when the absolute of the eigenvalue, i.e. $|\lambda| = 1$, $J_i(\lambda)^N$ has a polynomial
form of $N$. This result can be generalized to all Jordan blocks of $J$. As shown in the proof of Theorem 1, this corresponds to the case when $G$ belongs to the polynomial class $^3$. Denote the second largest eigenvalue of $T$ as $b$, then $\beta \cdot b^N$ dominates $m_p$, where $\beta$ is some constant. As such, one can easily derive the proof by following the proof for the exponential class in Theorem 1.

Theorem 2 indicates that the entropy of an RG lies in the spectrum of its associated DFA. Specifically, in the polynomial and exponential classes, a DFA with its summed transition matrix $T$ having only one diagonal element that is equal to 2 indicates that this DFA has only one “absorbing” state (either the accepting or rejecting state). Assume that a DFA has one absorbing state and is running over a string. Once reaching the absorbing state, this DFA makes a permanent decision – either acceptance or rejection – on this string, regardless of the ending symbol has been read or not. In comparison, in the proportional class, a DFA can have either zero or two absorbing states (one accepting state and one rejecting state).

In the case of Tomita grammars, every grammar has exactly one absorbing state except for grammar 5 and 6, which have no absorbing states. The DFAs for grammars 5 and 6 can only determine the label of a string after processing the entire string. Here it is not difficult to generalize the above-mentioned classification Theorems 1 and 2 to regular grammars with an arbitrary alphabet. Specifically, given a regular grammar $G$ with a $I$-size alphabet $\Sigma_I$ and its associated minimal DFA $M$ with $n$ states, let $T_i \in \mathbb{Z}^{n \times n}$ denote the transition matrix of $M$ associated with $i$th input symbol for $i \in \{1, \ldots, I\}$, and $T = \sum_i T_i$ is the sum of all transition matrices. We use $k(T)$ to represent the number of diagonal elements equal to $I$ and $\sigma(T)$ to denote the set of modulus of all eigenvalues of $T$. $|\lambda_2|$ is used to represent the second largest modulus of the eigenvalues of $T$. Then its categorization result for the three classes introduced in Theorem 1 and 2 is shown in the following table (without loss of generality here we only derive the result using the number of positive samples, i.e., $m_p$, $^3$We omit the proof since the number of diagonal elements of $T$ equal to 2 is 1 as discussed in the paper.

### Table 3.1 Generalization to the Regular Grammars with an Arbitrary Alphabet Size.

<table>
<thead>
<tr>
<th>Category</th>
<th>Data-Driven Perspective</th>
<th>DFA Perspective</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>$m_p \sim P(N)$, polynomial of $N$</td>
<td>$k(T) = 1$ and $\sigma(T) = {1, I}$</td>
<td>$H(G) = 0$</td>
</tr>
<tr>
<td>Ex</td>
<td>$m_p \sim \beta \cdot b^N$, $b &lt; I$ and $\beta &gt; 0$</td>
<td>$k(T) = 1$ and $\sigma(T) - {1, I} \neq 0 \Rightarrow H(G) = \log_I</td>
<td>\lambda_2</td>
</tr>
<tr>
<td>Pr</td>
<td>$m_p \sim \alpha \cdot I^N$, $\alpha \in (0, 1)$</td>
<td>$k(T)$ = 0 or $k(T) = 2$</td>
<td>$H(G) = 1$</td>
</tr>
</tbody>
</table>
defined by this grammar).

Recall that we require that for all RGs, the set of accepted strings has a smaller cardinality. As such, the incomplete DFA for a given RG can be obtained by deleting the rejecting absorbing state \(^4\) and all the edges associated with the state from a minimal complete DFA. Similarly, the transition matrix \(\hat{T}\) of the incomplete DFA becomes:

**Corollary 2.** Given any regular grammar \(G\) with \(\Sigma = \{0, 1\}\), we have

(a) \(G\) belongs to \([\text{Po}]\) if and only if \(L(\hat{T}) = 1\);
(b) \(G\) belongs to \([\text{Ex}]\) if and only if \(\left| L(\hat{T}) \right| \in (1, 2)\) and \(H(G) = \log_2 \left| L(\hat{T}) \right| ;\)
(c) \(G\) belongs to \([\text{Pr}]\) if and only if \(L(\hat{T}) = 2\).

Here \(L(\hat{T})\) denotes the largest eigenvalue of \(\hat{T}\).

Corollary 2 together with Theorem 2 provides a complete analysis of classification results for a DFA representation. It is easy to see that neither strictly local nor strictly piecewise belongs to the proportional class since they all have one absorbing-rejecting state. They can be categorized into either the polynomial or exponential class according to their specific forbidden factors or subsequences.

### 3.2.4 Generating Function Representation

Now we derive the classification results based on the generating function representation of a regular grammar. A generating function encodes an infinite sequence of numbers by treating them as the coefficients of a power series and is widely applied in combinatorial enumeration. Specifically, the generating function of regular grammar is defined as follows:

\[
f(x) = \sum_{N} m_p^N \cdot x^N, \quad \text{(3.5)}
\]

An important result is that generating function of regular grammar is a rational function. We have the following theorem for classification:

**Theorem 3.** Given any regular grammar \(G\) with \(\Sigma = \{0, 1\}\), we have

(a) \(G\) belongs to \([\text{Po}]\) if and only if \(\{m_p^N\}\) is finite or the radius of convergence \(r\) of \(f(x)\) is

\(^4\)This is important since, for a proportional class, there might exist two absorbing states and the accepting absorbing state in that case cannot be deleted. A more significant result is that in the monoid representation of regular grammar, only the monoid associated with polynomial and exponential grammar is a nullloid. The discussion of monoid representation is out of scope here.
equal to 1;
(b) $G$ belongs to $[\text{Ex}]$ if and only if the radius of convergence $r$ of $f(x)$ is between $1/2$ and 1, and $H(G) = -\log_2 r$;
(c) $G$ belongs to $[\text{Pr}]$ if and only if the radius of convergence $r$ of $f(x)$ is equal to 1/2.

The theorem can be understood from two perspectives. First, from Theorem 1 we can readily derive the radius of convergence by a ratio test. Another is related to calculating the generating function from a DFA. Specially, we have the following lemma [77]:

**Lemma 2.** The generating function $f_{ij}(x)$ from state $i$ to state $j$ is given by

$$f_{ij}(x) = \frac{(-1)^{i+j} \det(I - xT : j, i)}{\det(I - xT)}, \quad (3.6)$$

where $(B : j, i)$ denotes the matrix obtained by removing the $j$-th row and $i$-th column of $B$.

Note that the radius of convergence $r$ only depends on the denominator of the function $f(x)$, and $r$ is the smallest pole of the function. By lemma 2, the denominator has the form $\det(I - xT)$. As such, the radius is the inverse of the largest eigenvalue of the transition matrix, which completes the proof of the Theorem 3. The classification theorem is a generating function of a regular grammar and can be easily generalized to a regular grammar with multiple alphabets.

### 3.2.5 Regular Expression Representation

Here we provide an analysis from a regular expression perspective. First, we consider the following interesting question: given a regular grammar, how many parameters one needs to uniquely determine a positive string? The question is closely related to automatically generate positive samples by computer. We start with a simple example. For Tomita grammar 7, we have $G = 0^*1^*0^*1^*$, we only need four numbers $a, b, c, d$ to generate a string $0^a1^b0^c1^d$. However, for a more complicated example Tomita grammar 4, we have $G = (\epsilon + 0 + 00)(1 + 10 + 100)^*$; in this case, we have to record both the number and location of the suffix string 1, 10, 100 separately. Hence we need $\mathbb{Z}^3$ numbers to generate a sample. Moreover, we have the following fact:

**Fact 1.** Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have
(a) $G$ belongs to $[\text{Po}]$ if and only if $N(G) = k$;
(b) $G$ belongs to $[\text{Ex}]$ if and only if $N(G) = \mathbb{Z}^k$;
(c) $G$ belongs to $[\text{Pr}]$ if and only if $N(G) = \mathbb{Z}^{\mathbb{Z}}$.

Here $k$ is a constant and $N(G)$ denotes the required number of parameters.

Another perspective to explore this problem is to apply group theory. Specifically, given a regular grammar, we can consider it as a topological space. Furthermore, it can be decomposed into several disjoint orbits of different base points, and the whole topological space is generated by the actions on these base points. In this way, these actions form a monoid, and the cardinality of the monoid reflects the complexity of the grammar. Again, we will use Tomita grammar 4 and 7 for example to illustrate this idea. For $G = 0^*1^*0^*1^*$, the base point of the topological space is $\epsilon$ and we define the following different actions:
- $h_1$–adding 0 in the first slot;
- $h_2$–adding 1 in the second slot;
- $h_3$–adding 0 in the third slot;
- $h_4$–adding 1 in the fourth slot. Note that first adding 1 in the second slot then adding 0 in the first slot is exactly the same as first adding 0 in the first slot then adding 1 in the second slot, which means that these actions commute to each other. Hence this monoid is an abelian monoid $^5$ and the cardinality of the monoid is $\mathbb{Z}^4$. For $G = (\epsilon + 0 + 00)(1 + 10 + 100)^*$, the base points are $\epsilon$, 0 and 00, and we define the following actions: $h_1$–attaching 1 in the back; $h_2$–attaching 10 in the back; $h_1$–attaching 100 in the back. Note that this monoid $H$ is no longer commutative since $h_1h_2$ acting on base point 0 gives the string 0101 while $h_2h_1$ acting on base point 0 gives the string 0011. Hence, in this case, the monoid is a free monoid generated by $h_1, h_2, h_3$ and the cardinality of this monoid is $3^\mathbb{Z}$. More generally, we have the following fact:

**Fact 2.** Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have
(a) $G$ belongs to $[\text{Po}]$ if and only if $C(H) = k$ or $C(H) = \mathbb{Z}^k$;
(b) $G$ belongs to $[\text{Ex}]$ if and only if $C(H) = k\mathbb{Z}$;
(c) $G$ belongs to $[\text{Pr}]$ if and only if $C(H) = k^\mathbb{Z}$.

Here $k$ is a constant and $C(H)$ denotes the cardinality of the monoid of actions on the topological space.

These facts 1 and 2 give the following theorem:

**Theorem 4.** Given any regular grammar $G$ with $\Sigma = \{0, 1\}$, we have
(a) $G$ belongs to $[\text{Po}]$ if and only if there is no $+$ inside the Kleene star $*$ in the simplest

\footnote{More formally, this monoid can be considered as the quotient of a free monoid by a normal submonoid, i.e., $H = F\langle h_1, h_2, h_3, h_4 \rangle / \langle h_i h_j h_i^{-1} h_j^{-1} \rangle, \forall i \neq j$.}
regular expression;
(b) \( G \) belongs to \([\text{Ex}]\) if and only if there exists a polynomial grammar inside the Kleene star * in the simplest regular expression;
(c) \( G \) belongs to \([\text{Pr}]\) if and only if there exists an exponential grammar inside the Kleene star * or \((0 + 1)^k\) in the simplest regular expression, where \(k\) is a constant.

Note that another perspective to understand Theorem 4 is to consider the connection between a regular expression and the corresponding generating function. Given a regular expression, one can get the generating function by replacing \(\emptyset\) by 0, \(\epsilon\) by 1, any symbol with \(x\), concatenation with multiplication, union with plus and Kleene star with \(1/(1 - f(x))\), respectively. Then we can apply Theorem 3 to derive the same result. Theorem 4 provides an analysis based on the regular expression. Note that except for the special case in proportional class, this result does not depend on the size of the alphabet. More examples are in the next section.

3.3 Case Study for Tomita Grammars

As our goal is to study the characteristics of subclasses of regular grammar in terms of machine learning, we use the often cited Tomita grammars [78] as examples for our analysis, which is consistent with the evaluation. For each classification result in the previous section 3.2, we will choose an example to illustrate its application. The full results for the Tomita grammars are listed in Table 3.2.

3.3.1 Set Representation

Here we use Tomita grammar 3 to illustrate the application of Theorem 1. Note that from Table 3.2, we can see that it is difficult to derive an explicit formula for \(m_p^N\) from a pure combinatorial approach. Instead we provide a brief proof that grammar 3 belongs to exponential class. Let \(G^N\) denotes the set of positive samples with length \(N\), and it is easy to notice that it can be decomposed into the following four disjoint sets: samples ending with consecutive even numbers of 1’s, samples ending with consecutive odd numbers of 1’s, samples ending with 0’s and the number of 0’s at the end limited by the constraint, samples

\[ \beta = \frac{1/3 \cdot (19 + 3\sqrt{33})^{1/3} + 1/3 \cdot (19 - 3\sqrt{33})^{1/3} + 1/3}{(586 + 102\sqrt{33})^{2/3} + 4 - 2(586 + 102\sqrt{33})^{1/3}} \] This is also known as a Tribonacci number.
**Table 3.2** Analysis of Tomita Grammars.

<table>
<thead>
<tr>
<th>$G$</th>
<th>$m_p^N$</th>
<th>$L(T)$</th>
<th>$f(x)$</th>
<th>Regular Expression</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{1-x}$</td>
<td>$1^*$</td>
<td>Po</td>
</tr>
<tr>
<td>2</td>
<td>0.5 + 0.5(-1)^N</td>
<td>1</td>
<td>$\frac{1}{1-x^2}$</td>
<td>$(10)^*$</td>
<td>Po</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>1.77</td>
<td>$\frac{(1+x)^x}{1-2x^2-2x^4}$</td>
<td>$0^<em>(11)^+0^</em>+1(11)^<em>(00)^</em>+(1(11))^*$</td>
<td>Ex</td>
</tr>
<tr>
<td>4</td>
<td>$\beta \cdot b^N$</td>
<td>1.84</td>
<td>$\frac{1+x+x^2}{1-x-x^2}$</td>
<td>$(\epsilon + 0 + 00)(1 + 10 + 100)^*$</td>
<td>Ex</td>
</tr>
<tr>
<td>5</td>
<td>$2^{N-2}(1 + (-1)^N)$</td>
<td>2</td>
<td>$\frac{1-2x^2}{1-4x^2}$</td>
<td>$(00 + 11)^<em>((01 + 10)(00 + 11)^</em>$</td>
<td>Pr</td>
</tr>
<tr>
<td>6</td>
<td>$(2^N - 2(-1)^N)/3$</td>
<td>2</td>
<td>$\frac{1-x}{(1-2x)(1+x)}$</td>
<td>$(10 + 01)^<em>((1 + 00)(01 + 10)^</em>$</td>
<td>Pr</td>
</tr>
<tr>
<td>7</td>
<td>$(N^3 + 5N + 6)/6$</td>
<td>1</td>
<td>$\frac{1-2x+2x^2}{(1-x)^3}$</td>
<td>$0^*1^*0^<em>1^</em>$</td>
<td>Po</td>
</tr>
</tbody>
</table>

ending with 0’s and the number of 0’s at the end not limited by the constraint. We use $A_1^N$, $A_2^N$, $A_3^N$, $A_4^N$ to denote these sets respectively. Hence we have that $m_p^N = A_1^N + A_2^N + A_3^N + A_4^N$. Furthermore, we have the following recursion formulas:

$$
A_1^N = A_2^{N-1}, \\
A_2^N = A_1^{N-1} + A_3^{N-1} + A_4^{N-1}, \\
A_3^N = \sum_{i=1}^{[(N-1)/2]} A_2^{N-2i}, \\
A_4^N = A_4^{N-1} + A_4^{N-1},
$$

where $[\cdot]$ denotes the floor function and the initial condition is $A_1^0 = 0$, $A_2^0 = 1$, $A_3^0 = 0$, $A_4^0 = 1$. By adding these formulas we have $m_p^N = m_p^{N-1} + m_p^{N-2} + A_2^{N-2}$, which implies that this sequence increases faster than Fibonacci sequence. Hence it cannot belong to the polynomial class.

In contrast, let $Z_1$ denote the event that an even number of 0’s followed by an odd number of 1’s, and $Z_2$ denotes the event that even number of 0’s always followed by an odd number of 1’s in the whole string. Hence when the length of the string $N$ approaches infinity, the expectation of the probability of $Z_2$ is an infinite product of the probability of $Z_1$, which approaches zero. The indicates that the ratio of positive samples approaches zero when $N$ approaches infinity. Thus $m_p^N$ does not contain the term $\alpha \cdot 2^N$. Finally, we conclude that
this grammar belongs to the exponential class.

3.3.2 DFA Representation

Here we use Tomita grammar 2 to illustrate the application of Theorem 2. The transition matrix of the complete DFA is $T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix}$. By simple calculation we have $k(T) = 1$ and $\sigma(T) = \{1, 2\}$. Also, the transition matrix of the incomplete DFA is $\hat{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and the largest eigenvalue is 1, which means it belongs to the polynomial class.

![Figure 3.3](image)

**Figure 3.3** Example for the Tomita-2 grammar where red (black) states are the accept (reject) states.

3.3.3 Generating Function Representation

Here we use Tomita grammar 4 to illustrate the application of Theorem 3. The generating function is $f(x) = 1 + x + x^2 - x - x^2 - x^3$, and the radius of convergence is the smallest positive pole of the function. Namely, we need to solve the equation $x^3 + x^2 + x - 1 = 0$, and we have $r = 0.544$, which lies between the values of 0 and 1. Hence it belongs to the exponential class.

3.3.4 Regular Expression Representation

Here we use Tomita grammar 5 to illustrate the application of Theorem 4. For $G = (00 + 11)^* (01 + 10)(00 + 11)^* (01 + 10)(00 + 11)^*$, there exists an expression $G' = (01 + 10)(00 + 11)^*$.
11)∗(01 + 10)(00 + 11)∗ inside the Kleene star. For grammar \( G' \) there exists a polynomial grammar \( 00+11 \) inside the Kleene star, indicating that \( G' \) is an exponential grammar. Hence \( G \) belongs to the proportional class.

### 3.4 Evaluation

Here our empirical results show that the categorization of RGs is related to the difficulty of RNNs to learn these grammars, and implementation is publicly available \(^7\). We evaluated three Tomita grammars \( G-1, G-3, G-5 \). For each grammar, its subclass is shown in Table 3.2. Following prior work [33], we generated three training sets of binary strings with their lengths ranging from 1 to 14 for these grammars. We also collected three validation sets of strings with different lengths of \([1, 4, 7, \ldots, 25]\) to ensure that the models can be trained to generalize to longer strings. The training process was terminated either when the model achieved a \( F_1 \)-score that is equal to or higher than 0.95 or when a maximum of 5000 epochs was reached.

We selected several different recurrent networks to demonstrate how the difficulty of learning generalizes to different models. Specifically, we trained SRN, 2-RNN, GRU, and LSTM with data generated on each grammar to perform a binary classification task. We configured all RNNs to have the same size of the hidden layer across all grammars and trained them on each grammar for 10 random trials using a mean squared error loss. In each trial, we randomly initialized the hidden layer of a model.

We followed previous research [27] and used either activation functions – \texttt{sigmoid} and \texttt{tanh} – to build these RNNs. Also, for each RNN, we used one-hot encoding to process the input alphabets of 0’s and 1’s. With this configuration, the input layer is designed to contain a single input neuron for each symbol in the target language alphabet. Thus, only one input neuron is activated at each time step. Moreover, we followed the approach introduced in previous research and applied the following loss function to all RNNs:

\[
L = \frac{1}{2}(y - h_0^T)^2.
\]

This loss function can be viewed as selecting a special “response” neuron \( h_0 \) and comparing it to the label \( y \), i.e., 1 for acceptance and 0 for rejection. Thus, \( h_0^T \) indicates the value of the neuron \( h_0 \) at time \( T \) after an RNN receives the final input symbol. By using this simple

\(^7\) https://github.com/lazywatch/rnn_theano_tomita.git
loss function, we attempt to eliminate the potential effect of adding an extra output layer and introducing more weight and bias parameters. Also, by this design, we ensure that the knowledge learned by an RNN resides in the hidden layer and its transitions. When applying different activation functions, we make sure that $h_0$ is always normalized between the range.
of 0 and 1, while other neurons have their values between 0 and 1 for sigmoid activation, and -1 to 1 for tanh activation. During training, we optimized parameters through stochastic gradient descent and employed RMSprop optimization [79].

Results are shown in Figure 3.4, where the results from all trials fit the shaded area associated with each plot. The $x$-axis represents the number of epochs during training, and the $y$-axis represents the loss. In figures 3.4a, 3.4b, 3.4c and 3.4d, we see that for $G$-1 and $G$-3 which have lower entropy values have learning that converges much faster and more consistently than that of $G$-5, which has the highest entropy value. This effect holds for all RNNs evaluated. Note that $G$-5 defines two sets of strings with equal cardinality when the string length is even. In this case, by flipping any binary digit of a string to its opposite (e.g., flipping a 0 to 1 or vice versa), a valid or invalid string can be converted into a string with the opposite label. This implies that a model must pay equal attention to any string to learn the underlying grammar, which makes the learning process more challenging.

To better illustrate the difference of the difficulty of learning $G$-1 and $G$-3, a zoomed view is provided in figures 3.4e, 3.4f, 3.4g and 3.4h for each plot at the top row of Figure 3.4. While the learning process of all models converges within 100 epochs for both grammars, it is clear that the learning process is slower for $G$-3. These results agree with both our analysis of the entropy of these grammars and our intuition.

It would be interesting to further investigate the relationship between recurrent models and grammar learning, which will be briefly discussed in the next section 3.5. A promising approach would be to investigate the connection between RNN and DFA representations since both of them are stateful models. In general, while it is possible to validate the connection between nonlinear RNNs and DFAs empirically, it has been challenging to establish a theoretical connection between nonlinear RNNs and finite state machines. Specifically, second-order RNNs naturally fit into the task of learning any DFA, while some first-order RNNs only represent a portion of DFAs, indicating that the gap (not computationally) between first-/second-order RNNs is not as significant as expected. However, we only observe the differences by experiments for these gated models (LSTM/GRU). To better study this, we discuss the following two questions: Given an RG, do there exist some common substrings misclassified by different RNNs? Given an RNN, are there consistent, persistent flaws when learning different RGs? Our results observed on the Tomita grammars can be considered an initial attempt to answer the above questions. We first find that all RNNs perform perfectly on polynomial RGs (e.g., G-1/2/7), and first-order RNNs perform poorly on proportional
RGs (e.g., G-5/6). Also, for first-order RNNs, their overall classification performance and misclassified strings indicate that these RNNs randomly choose some RGs to learn when learning proportional RGs. For exponential RGs (e.g., G-3), we find some patterns in strings misclassified by certain first-order and gated RNNs. For example, LSTM tends to ignore "101" appearing at the end of a string and subsequently has high false-positive errors (and learns fewer deterministic transitions). In contrast, the SRN tends to lose a count of consecutive 0s or 1s. These results indicate that these RNNs learn a smoothed boundary of the manifold that holds these strings. Since these common substrings are more likely to lie in the periphery of the manifold, which suggests the use of up-sampling to compensate for this problem.

3.5 Discussion

In this section, we briefly discuss the proposed definition of entropy and also the relationship between the metric entropy and another complexity measure – average edit distance. At last, we establish the connection between different RNNs and the aforementioned different classes of regular grammars.

3.5.1 Different Definitions of Entropy

The proposed entropy measures the expected times of two consecutive strings having different labels when assuming each string is arranged randomly as a point along a line with the length of $|X^N|$. This entropy is naturally more informative than that defined in shift space since it considers both the strings accepted and rejected by a DFA. However, this entropy only considers strings with a fixed length of $N$. As such, it can be taken as an analogy of calculating the entropy of an isolated subsystem (consists of strings in $X^N$) which is only part of a microscopic system (consists of strings in $X^{\leq N}$). To introduce a more general form of the entropy for regular grammars, we provide the following proposition in a semi-microscopic perspective:

**Proposition 2.**

$$H(G) = 1 + \lim_{N \to \infty} \frac{1}{N} \log_2 T_N. \quad (3.8)$$

where $T_N = \sum_{k=1}^{N} m_p^k m_n^k / 2^{k+N}$. 

38
This proposition can be considered as a corollary of Theorem 1, and one can easily verify that the eq. (3.8) holds. In comparison with eq. (3.1), the right-hand side of eq. (3.8) represents the sum of expectations of having two consecutive points with different labels as the length of the underlying line increases from 1 to \(N\). In other words, eq. (3.8) measures the complexity of a regular grammar, of which the associated set of strings is treated as a microscopic system from a macro perspective. As such, we can take Proposition 2 as an alternative definition of \(H(G)\) but with clearer physical meaning. However, both eq. (3.1) and eq. (3.8) are difficult to compute in practice. In the following, we propose another definition for the entropy of regular grammar with a binary alphabet in a macroscopic perspective. This definition possesses proper physical meaning while requiring easier computation.

**Definition 4.** The entropy of a regular grammar \(G\) with alphabet \(\Sigma = \{0, 1\}\) is defined as:

\[
\hat{H}(G) = 1 + \lim_{N \to \infty} \frac{1}{N} \log_2 (r_p \cdot r_n).
\] (3.9)

In our Definition 4, we avoid using \(\lim \sup\) as it could complicate the calculation. Note that we have \(H(G) = \hat{H}(G)\), which will be discussed in 3.5.2. This facilitates us to estimate the entropy of grammar with finite strings by eliminating the restriction on the length of strings. For example, given a data set with a fair amount of strings sampled from an unknown data set, we can estimate the complexity of this unknown data set by calculating the entropy of the available data set. Furthermore, similar to eq. (3.8), Definition 4 has a more general form as it covers all strings in \(X \leq N\), hence preserves the property of measuring the complexity of a regular grammar from a macro perspective.

**3.5.2 Connecting Entropy with Average Edit Distance**

In a previous study [24], an extension of the *edit distance* was defined to measure the difference between two sets of strings with a certain length. This previous definition adapts edit distance from measuring the difference between different grammars to measuring the complexity of a certain grammar. It has been demonstrated that a regular grammar with a smaller extended edit distance will be more challenging for robust learning [24, 59]. Here, we further extend the definition mentioned above to a more general form, which measures

---

8Here we do not consider the empty grammar and \((0 + 1)^*\).

9*Edit distance, or Levenshtein distance, measures the minimum number of operations – insertion, deletion, and substitution of one symbol from a string – needed to convert a string into another [10].
the difference between two sets of strings with varying lengths. We denote the edit distance between two strings \( x, x' \in X^{\leq N} \) by \( d_e(x, x') \) and the edit distance between \( x \) and a set \( A \) by \( \hat{d}_e(x, A) = \min_{x' \in A} d_e(x, x') \). Then we have:

**Definition 5 (Average Edit Distance).** The average edit distance \( D(G) \) of a regular grammar \( G \) is defined as:

\[
D(G) = \frac{1}{2} \cdot \lim_{N \to \infty} \left( \frac{1}{m_p} D_p + \frac{1}{m_n} D_n \right),
\]

where \( D_p = \sum_{x \in X_p^{\leq N}} \hat{d}_e(x, X_n^{\leq N}) \) and \( D_n = \sum_{x \in X_n^{\leq N}} \hat{d}_e(x, X_p^{\leq N}) \).

In comparison, the previously mentioned extended edit distance [24] is defined as:

\[
\hat{D}(G) = \frac{1}{2} \cdot \liminf_{N \to \infty} \left( \frac{1}{m_p} D_p^N + \frac{1}{m_n} D_n^N \right),
\]  

where \( D_p^N = \sum_{x \in X_p^N} \hat{d}_e(x, X_n^N) \) and \( D_n^N = \sum_{x \in X_n^N} \hat{d}_e(x, X_p^N) \).

Note that different from the proposed definitions of entropy, these two definitions of average edit distance are not equivalent. Specifically, given a string \( x \in X_p^N \), \( \hat{d}_e(x, X_n^{\leq N}) = \hat{d}_e(x, X_n^N) \) does not hold in general. In other words, the string in \( X_n^{\leq N} \) that minimizes the edit distance between \( x \) and \( X_n^{\leq N} \) does not necessarily have a length of \( N \). For example, consider a regular grammar \( G \), which defines binary strings with lengths that are divisible by 3. It is easy to see that \( D(G) = 1 \) while \( \hat{D}(G) = 0 \). As such, eq. (3.10) is more informative than eq. (3.11).

**Connecting Entropy with Average Edit Distance** Note that our newly defined measures maintain the classification results of regular grammars but with a more rigorous setting and convenient calculation. By comparing the entropy and average edit distance defined previously, it is evident that these two defined metrics are closely related to each other. More formally, we have the following theorem that highlights the connection between the average edit distance and the entropy:

**Theorem 5.**

\[
H(G) = \lim_{N \to \infty} \frac{1}{N} \cdot \log_2(r_p D_n + r_n D_p) = \limsup_{N \to \infty} \frac{1}{N} \cdot \log_2(r_p D_n^N + r_n D_p^N) = \hat{H}(G).
\]
Proof. It is easy to check that
\[
0 \leq \limsup_{N \to \infty} \frac{1}{N} \log_2 D^N(G) \leq \limsup_{N \to \infty} \frac{1}{N} \log_2 N = 0. \tag{3.13}
\]
As such we have
\[
\limsup_{N \to \infty} \frac{1}{N} \log_2 (r_p D^N_n(G) + r_n D^N_p(G)) \\
= \limsup_{N \to \infty} \frac{1}{N} \log_2 \left( \frac{r_p D^N_n(G) + r_n D^N_p(G)}{2 r_p r_n G^2 N} \right) (2 r_p r_n G^2 N) \\
= \limsup_{N \to \infty} \frac{1}{N} \left( \log_2 (2 r_p r_n G^2 N) + \log_2 D^N(G) \right) \\
= \hat{H}(G) + \limsup_{N \to \infty} \frac{1}{N} \log_2 D^N(G). \tag{3.14}
\]
Where \( D^N(G) = \frac{1}{m_p} D^N_p(G) + \frac{1}{m_n} D^N_n(G) \). Since the sequence \( \lim_{N \to \infty} \frac{1}{N} \log_2 D^N(G) \) is bounded and converges to zero, we have \( \hat{H}(G) = \limsup_{N \to \infty} \frac{1}{N} \log_2 (r_p D^N_n(G) + r_n D^N_p(G)) \). Similarly, the other side of equation of \( H(G) \) holds by the same argument. To see \( H(G) = \hat{H}(G) \), again we apply Theorem 1. The summation will give the same expression in each subclass, which can be verified by simple calculation. In this way, the limit exists when we consider all samples with length not greater than \( N \). Hence we have \( H(G) = \hat{H}(G) \).

By assuming a random variable \( \tilde{D} \) that takes a value from \( \{D_n, D_p\} \) with the probability of \( r_p \) for selecting \( D_n \) and \( r_n \) for selecting \( D_p \), then Theorem 5 calculates \( \mathbb{E}[\tilde{D}] \), which is the expected summation of the edit distance between positive and negative strings for any grammar \( G \). Furthermore, let \( H_N(G) = \frac{1}{N} \log_2 (r_p \cdot r_n) \) and \( D_N(G) = \frac{1}{m_p} D_p + \frac{1}{m_n} D_n \). It is clear that \( D_N(G) \) conveys more information than \( H_N(G) \), since the former concerns not only the cardinality of the set of strings but also the distance between different sets of strings. For example, consider the following two regular grammars: \( G_1 = 1^*(0 + 1) \) and \( G_2 = 1^* + 0^* \). It is easy to check that these two grammars have the same \( H_N \). However, their corresponding
average edit distance values are rather different:

\[ D_N(G_1) = 1 + \frac{2^{N-1}(N-1)}{2^N-2}, \]

\[ D_N(G_2) = 1 + \frac{1}{2^N-2} \left( \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} i \binom{N}{i} + \sum_{i=1}^{\left\lceil \frac{(N-1)}{2} \right\rceil} i \binom{N}{i} \right). \]  \hfill (3.15)

It can also be noticed that as \( N \) approaches infinity, both of \( D_N(G_1) \) and \( D_N(G_2) \) also approach infinity. This raises the following conjecture:

**Conjecture 1.** The mapping from \( H(G) \) to \( D(G) \) is a bijection \( f : [0, 1] \to [1, \infty] \).

### 3.5.3 Additional Comments on Grammar Learning

Grammar learning does not only depend on the complexity of grammars but also the RNN structures. Here we briefly discuss the role of RNNs in grammar learning in terms of their expressive power. Specifically, we show the equivalence between the linear 2-RNN and DFA. Then we examine if other RNNs can learn DFAs with different levels of complexity and proposed a unified framework for recurrent models. Here we only consider RNNs with hidden linear activation for analytical convenience.

#### Linear 2-RNN and DFA Relationship

Given a DFA with an \( I \)-size alphabet \( \Sigma_I \) and a minimal number of \( n \) states, we denote the transition matrix for each input symbol as \( T_i \in \mathbb{Z}^{n \times n} \). Each column of \( T_i \) sums to 1. Given an input symbol \( i \in \Sigma_I \), the DFA state transition is \( h^t = T_i h^{t-1} \), where \( h^t \) denotes the hidden vector at time \( t \) in the \( n \)-dimensional unit cube \( H = [-1, 1]^n \). Assume the input of a linear 2-RNN is one-hot encoded, then a

<table>
<thead>
<tr>
<th>RNN (Linear)</th>
<th>RG</th>
<th>Solutions</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-RNN</td>
<td>All</td>
<td>( W_i = T_i, i = 1, \ldots, I )</td>
<td>( W' = W, U' = 0, V' = 0, b' = 0 )</td>
</tr>
<tr>
<td>SRN</td>
<td>Grammars accept strings with certain length.</td>
<td>( V = \sum_{i=1}^I a_i T_i, c_i = 0, \sum_i a_i = 1, a_i &gt; 0 )</td>
<td>( W' = 0, U' = U, V' = V, b' = b )</td>
</tr>
<tr>
<td>MI-RNN</td>
<td>Grammars accept strings with certain length.</td>
<td>( V = \sum_{i=1}^I a_i T_i, U = \mathbb{1}^{n \times 1}, \sum_i a_i = 1, a_i &gt; 0 )</td>
<td>( W'<em>{ijk} = a_i U</em>{ik} V_{ij}, b' = b, U' = \beta_2 U, V' = \beta_1 V )</td>
</tr>
</tbody>
</table>

Table 3.3 Solutions for approximating DFA with 2-RNN, Elman-RNN, and MI-RNN.
linear 2-RNN can be constructed that exactly matches the DFA by solving:

$$\min \limits_{W_i \in \mathbb{R}^{n \times n}} \int_\mathcal{H} \sum_ {i=1}^I |T_i h - W_i h| \, d\mu(h).$$  \hspace{1cm} (3.16)

where \(\mu(h)\) denotes the probability measure of the random variable which reflects the distribution of individual points in the hidden space. The optimum is obtained when \(W_i = T_i\) for \(i = 1, \ldots, I\). Although this optimum is challenging to reach when \(T_i\) is not available, (3.16) indicates that a 2-RNN can be stably constructed to resemble any DFA [80, 81]. As for M-RNN, similar results can be obtained if its adopted decomposition retains a significant fraction of the tensor’s expressive power [82].

**Correspondence Between SRN, MI-RNN and DFA**  Here we first fit the reformulated linear SRN and linear MI-RNN into the optimization framework introduced for linear 2-RNN. Specifically, for SRN, we use \(c_i = U x_t + b\) (shown in Table 2.1) to represent the input-dependent term. We assume \(h\) is uniformly distributed in \(\mathcal{H}\). Then we have:

$$\min \limits_{V \in \mathbb{R}^{n \times n}, c_i \in \mathbb{R}^{n \times 1}} \int_\mathcal{H} \sum_ {i=1}^I |T_i h - V h + c_i| \, dh. \hspace{1cm} (3.17)$$

For MI-RNN, we only consider the \(h_t+1 = U x_t \odot V h_t\) term which dominates the transition and have The rest terms shown in Table 2.1 can be omitted since they do not contribute to the transition matrix. Then we have

$$\min \limits_{V \in \mathbb{R}^{n \times n}, U_i \in \mathbb{R}^{n \times 1}} \int_\mathcal{H} \sum_ {i=1}^I |T_i - (1_n \otimes U_i^T \odot V)| \, dh, \hspace{1cm} (3.18)$$

where \(1_n\) is the all 1 column vector, \(U_i\) is \(i\)-th column of \(U\). \(\otimes\) denotes the Kronecker product. The solutions for (3.17) and (3.18) are shown in the third column in Table 3.3. From (3.17), it is easy to check given a fixed \(V\), the optimum can be achieved when \(T_i = T_j\) for \(i, j = 1, \ldots, I\) and \(c_i = 0\). This indicates that SRN and MI-RNN can only accurately learn DFAs that recognizing strings with a certain length; therefore, they are limited in their capability of approximating all three classes of DFAs. Specifically, a DFA that only accepts strings with a certain length can either belong to the polynomial or exponential class. A similar result can be obtained for MI-RNN and will be discussed in the following.
An Unified View of Different RNNs  While linear 2-RNN is better at modeling RGs than MI-RNN and SRN, these latter models can be more suitable for modeling other types of sequential data. For example, prior work [83] shows that SRN generalizes nonlinear autoregressive-moving average models. As such, we now propose a unified framework that integrates different orders of hidden interaction, while preserving the advantages of different RNNs:

$$h^t = \phi\left(\sum_i W'_{ij}x_i^t h^{t-1} + U' x^t + V' h^{t-1} + b'\right),$$  

where $W' \in \mathbb{R}^{I \times n \times n}$.

![Figure 3.5](image-url)  

**Figure 3.5**  Recurrent model hierarchy.

We show in the fourth column of Table 3.3 about how to configure this unified RNN (UNI-RNN) to express SRN, MI-RNN, and 2-RNN. Specifically, for MI-RNN, we can see that in its associated unified framework, $W'_{ijk} W'_{imn} = W'_{ijm} W'_{ink}$. This indicates that for a pair of input symbols, their sequence order can be switched to reach the same state. This form of transition matrix corresponds to a DFA that only accepts strings with a certain length. This unified framework applies flexible control of the hidden layer of an RNN during the learning process. Specifically, the four terms on the right-hand side of (3.19) represent the input-dependent rotation and translation, input-independent rotation and translation, respectively. Other RNNs only partially support these transformations. Similar to the Chomsky hierarchy of formal grammars, we plot the recurrent model hierarchy in Figure 3.5 based on their representation capability. As such, we expect the UNI-RNN to be more flexible for modeling different types of sequential data.
3.6 Conclusion and Future Work

In this chapter, theoretical analysis and empirical validation for subclasses of regular grammars are presented. Specifically, to measure the complexity of regular grammar, we introduced an entropy metric based on the concentric ring representation, which essentially reflects the difficulty in training RNNs to learn the grammar. Using entropy, we categorized regular grammar into three disjoint subclasses: polynomial class, exponential class, and proportional class. In addition, we provided classification theorems for different representations of regular grammar. Given a regular grammar, these theorems use their corresponding properties in the given representation to efficiently determine which subclass a grammar belongs to without calculating the entropy value. Several common representations, including deterministic finite automata, regular expression, and sets have a corresponding case study that illustrates their applications. This categorization could also be applied to other relevant representations. Finally, we conducted an experiment to demonstrate the influence of grammar learning based on its complexity, which validates the effectiveness of the proposed entropy metric and the theoretical analysis. We believe this will provide a deeper understanding of the internal structure of the regular grammar in the context of learning. We also provided another perspective to study grammar learning, i.e., recurrent models. All RNNs except second-order RNN have problems learning certain classes of grammars. It would seem the grammar chosen matters more than the RNN architecture. Such a connection between DFA and RNNs can provide insights into explainable artificial intelligence and adversarial machine learning.
Chapter 4

Topological Analysis of Sample Influence for Neural Networks

Data samples collected for training machine learning models are typically assumed to be independent and identically distributed (iid). Recent research has demonstrated that this assumption can be problematic as it simplifies the manifold of structured data. This has motivated different research areas such as data poisoning, model improvement, and explanation of machine learning models. In this chapter, we study the influence of a sample on determining the intrinsic topological features of its underlying manifold. We propose the Shapley Homology framework, which provides a quantitative metric for the influence of a sample of the homology of a simplicial complex. More specifically, our proposed framework consists of two main parts: homology analysis, where we compute the Betti number of the target topological space, and Shapley value calculation, where we decompose the topological features of a complex built from data points to individual points. By interpreting the influence as a probability measure, we further define an entropy that reflects the complexity of the data manifold. Furthermore, we provide a preliminary discussion of the connection of the Shapley Homology to the Vapnik-Chervonenkis (VC) dimension. Empirical studies show that when the Shapley homology is used on neighboring graphs, samples with higher influence scores have a greater impact on the accuracy of neural networks that determine graph connectivity and on several regular grammars whose higher entropy values imply greater difficulty in being learned.
4.1 Introduction

In much machine learning research, it is common practice to assume that the training samples are independent and identically distributed (iid). As such, samples are implicitly regarded as having an equal influence on determining the performance of a machine learning model. Recently, limitations of this assumption have been explored. For example, a recent study [84] showed that certain training samples could have significant influences on a model’s decisions for certain testing samples. This effect has motivated research on model interpretation [15, 84, 85], model & algorithm improvement [86–88], and poisoning attacks [84, 89, 90].

Many of the aforementioned methods study the sample influence problem by using neural networks. Specifically, it is common to adopt a neural network (either a target model or a surrogate) to identify samples that are deemed as more influential [84, 87]. However, as shown by [90] in a poisoning attack scenario, a limited number of poisoning samples can be effective in applying a backdoor attack in a model-agnostic manner. This motivates work that studies the intrinsic properties of data so that one can develop countermeasures to methods that defeat learning models. Specifically, by representing the underlying space of data as a topological space, we study the sample influence problem from a topological perspective. The belief that topological features of the data space better represent its intrinsic properties has attracted recent research on establishing relationships between machine learning and topological data analysis (TDA) [91–93].

In this chapter, we consider TDA as a complementary approach to other approaches that study sample influence. We propose a framework for decomposing the topological features of a complex built with data points to individual points. We interpret the decomposed value for each point as representing its influence on affecting its topological features. We then calculate that influence as Shapley values [94]. Recent research [95–97] proposed similar approaches to quantify feature influence. By interpreting the influence as a probability measure defined on a data space, we also calculate the entropy, which describes the complexity of this space. Under our framework, we devise two algorithms based on different Betti numbers for calculating the influence of data samples and the entropy for any data set.

We perform both analytical and empirical studies on several sets of data samples from two families of structured data – graphs and strings. Specifically, we generate random graphs using the Erdos-Renyi model and binary strings with regular grammars. In both cases, we use neural networks as verification tools for our analysis of sample influence. Explicitly,
we employ a feed-forward network trained to determine the connectivity of the generated graphs and a recurrent network trained to recognize the generated strings. Our results show that samples identified by our method as having more significant influence indeed have more impact on the connectivity of their underlying graphs and that grammars with higher complexity have more difficulty in being learned.

4.2 Related Work

In this section, we review some related works on sample influence and topological data analysis.

4.2.1 Sample Influence

Recently, research has focused on how to explore and exploit influential samples. As an example, by analyzing how the performance of a model in the testing phase affects a small number of training samples motivated research in poisoning attacks [84,89,90]. In this case, a limited number of corrupted training examples are injected to degrade a target model’s performance. An alternative thread of research exploits this effect to improve the generalization accuracy of learning models and the efficiency of learning algorithms. Specifically, influential samples can be identified via learning and then used for enhancing a models’ performance on imbalanced and corrupted data [86,88]. They can also be synthesized to represent a much larger set of samples, thus accelerating the learning process [87]. In addition, influential samples can bring explainability to deep learning models by identifying representative samples or prototypes used in decision making [85,98].

Typical approaches measure the influence of a data point by decomposing the target function’s values with respect to all of the data points of their contributions [84]. More precisely, they observe the unit change of the value of the target function caused by this point. In most cases, the target function is the output of a trained model. We follow the same approach but change the decomposition rule and particularly the target function. In our case, instead of building a model-dependent framework, we reveal the internal dependencies between data in a metric space.
4.2.2 Topological Data Analysis (TDA)

Currently, the most widely-used tool in TDA is persistent homology [91], which is an algebraic method for measuring the topological features of shapes and functions. This method provides a multi-scale analysis of the underlying geometric structures represented by persistence barcodes or diagrams. Most of the previous research using persistent homologies has focused on the classification task by identifying manifolds constructed by data points [99–101] and mapping the topological signatures to machine learning representations [93, 102]. Unlike previous work, our proposed Shapley homology emphasizes revealing the influence of a vertex on the topological features of a space from a microscopic viewpoint and assembling individual influence from which to derive an entropy measure that reflects the macroscopic features of a space consisting of individual vertices.

4.3 Shapley Homology

Here, we introduce the relevant concepts that are necessary for our Shapley Homology framework. We then provide the definitions of sample influence and entropy.

4.3.1 Čech Complex and Homology Group

Čech Complex In topological analysis, the study of a set of data points is typically based on simplicial complexes, which provide more abstract generalizations of neighboring graphs that describe the structural relationships between data points. Here, we are particularly interested in the Čech complex, which is an abstract simplicial complex. More formally, the definition of the Čech complex is as follows.

Definition 6 (Čech complex). The Čech complex $C_r(X)$ is the intersection complex or nerve of the set of balls of radius $r$ centered at points in $X$.

In particular, given a finite point cloud $X$ in a metric space and a radius $r > 0$, the Čech complex $C_r(X)$ can be constructed by first taking the points in $X$ as a vertex set of $C_r(X)$. Then for each set $\sigma \subset X$, let $X \in C_r(X)$ if the set of $r$-balls centered at points of $X$ have a nonempty intersection. Note that the Čech complex has a property showing that the simplicial complex constructed with a smaller $r$ is a subcomplex of one constructed with a larger $r$. In this way, we can obtain a filtration.
The topological features of data space are typically formalized and studied through Homology, which is a classical concept in algebraic topology [91]. In the following, we briefly introduce homology to the extent that is necessary for understanding its role in our framework.

**Definition 7** (Homology Group). A chain complex \((A_n, d_n)\) is a sequence of abelian groups or modules connected by homomorphisms \(d_n : A_n \to A_{n-1}\), such that the composition of any two consecutive maps is the zero map. The \(k\)-th homology group is the group of cycles modulo boundaries in degree \(k\), that is,

\[
H_k := \frac{\ker d_n}{\text{im } d_{n+1}},
\]

where \(\ker\) and \(\text{im}\) denotes the kernel and image of the homomorphism, respectively.

Generally speaking, the \(k\)-th homology is a quotient group that indicates the \(k\)-dimensional independent features of the space \(X\). Particularly, when \(k = 0\), we have the following proposition [103]:

**Proposition 3.** For any space \(\mathcal{X}\), \(H_0(\mathcal{X})\) is a direct sum of abelian groups, one for each path-component of \(\mathcal{X}\).

Specially, when a complex takes the form of a neighboring graph, then the 0-th-homology denote the subgraphs that are connected in this graph. As for the 1-st-homology \(H_1\), it represents the number of 1-dimensional holes in the given manifold.

### 4.3.2 Sample Influence and Entropy

Given a point cloud represented by a Čech complex, we study the influence of each data point on the topological features of this complex. This influence of a data point can be further interpreted as the probability that a unit change of the topological feature is caused by this point. More formally, denote a data set containing \(n\) samples by \(X = \{x_1, \cdots, x_n\}\), we have the following definition of sample influence.

**Definition 8** (Sample Influence). Given a discrete space \((X, \Sigma)\), the influence of any sample (or subset) \(\{x\} \subset X\) is a probability measure \(\mu\).

We then define an entropy that shares a similarity to the entropy in information theory; both reflect the number of internal configurations or arrangements of a system.
Definition 9 (Entropy). *Given a probability measure μ, the entropy of a dataset X is defined as \( H(X) = - \sum_{i=1}^{n} \mu(x_i) \log \mu(x_i) \).*

When introducing new data points, these data points may change the number of boundaries needed to separate the data points in the given metric space. Moreover, each of the newly added data points will have a different influence on the resulting number. By interpreting the influence as a probability measure, our defined entropy quantifies the number of boundaries in a data set and thus implies the capacity that a neural network needs to learn this data set.
4.3.3 The Framework of Shapley Homology

Here we propose a framework (depicted in Figure 4.1) of Shapley Homology in order to study sample influence. Specifically, in Figure 4.1, we provide an example for investigating a specific topological feature – the Betti number [104] –of a topological space. As will become clear from the following, the Betti number can be used to quantify the homology group of topological space according to its connectivity.

**Definition 10** (Betti Number). Given a non-negative integer \( k \), the \( k \)-th Betti number \( \beta_k(\mathcal{X}) \) of the space \( \mathcal{X} \) is defined as the rank of the abelian group \( H_k(\mathcal{X}) \), that is,

\[
\beta_k(\mathcal{X}) = \text{rank}(H_k(\mathcal{X})).
\] (4.2)

Following the definition, as there are no special structures (such as real projective space in the complex), the Betti number \( \beta_k(\mathcal{X}) \) indicates the number of the direct sum of abelian groups of the \( k \)-th homology group. In other words, the \( k \)-th Betti number refers to the number of \( k \)-dimensional voids on a topological surface.

While the Betti number only indicates an overall feature of a complex, we need to further distribute this number to each vertex of this complex as its influence score. Recent research [95] on interpreting neural networks has introduced Shapley value from cooperative game theory to decompose a classification score of a neural network made for a specific sample to individual features as their importance or contribution to rendering this classification result. Inspired by this line of work, we also employ the Shapley value as the influence score for each vertex. However, it should be noted that for a fixed \( k \), the Betti number \( \beta_k \) does not satisfy the monotonicity property of Shapley values. That is, the Betti number of \( X_1 \) is not necessarily larger than that of \( X_2 \) when \( X_1 \) is a subcomplex of \( X_2 \). As a result, we cannot adopt the formulation for calculating Shapley values directly. Here we use the following variant formulation for calculating Shapley values for a Čech complex:

\[
s(x_i) = \sum_{C \subseteq \mathcal{X} \setminus x_i} \frac{|C|!(n - |C| - 1)!}{n!} |\beta(C \cup x_i) - \beta(C)|.
\] (4.3)

It is important to note that in our formulation (4.3), we use the absolute value to resolve the monotonicity issue.

It is clear that our formulation (4.3) satisfies the symmetry axiom, whereas it does not
satisfy other Shapley axioms, including the linearity and carrier axioms [94]. Nonetheless, our formulation still measures the marginal effect of a vertex. Besides, as we mainly focus on measuring the topological feature, both the decrease and increase in Betti number of a vertex are crucial for determining the influence. Furthermore, since our entropy is symmetry-invariant, its value will remain the same under the group action on the vertices [105].

The above discussion indicates that the influence of a data sample can be regarded as a function of the radius $r$, the Betti number $\beta_k$, the size of the data set containing this sample, and the topological space constructed upon the chosen metric. Unlike persistent homology [106], which studies the topological features of data space at varying “resolution” $r$, our analysis takes a more static view of the topological features of a complex built with a fixed $r$. As such, our analysis can be viewed as taking a slice from the filtration used for persistent homology. In the following section, we propose two algorithms based on Betti number $\beta_0$ and $\beta_1$ respectively for calculating the influence scores of data points in a Čech complex constructed with certain specified $r$ and $k$.

### 4.4 Algorithm Design and Case Study

Here we will focus on algorithm design based on zeroth Betti number $\beta_0$ and first Betti number $\beta_1$. Then we provide several case study on simple grammars and some conclusions of special families of graphs as well.

#### 4.4.1 Algorithm and case study of $\beta_0$

With Proposition 3 and Definition 10 and when the Betti number $\beta_0$ is regarded as a quantitative indicator, it equals the number of connected components in a complex. In this case, the task of calculating $\beta_0$ of a Čech complex is equivalent to calculating the number of connected components of a graph. This enables us to compute the Laplacian matrix $L$\footnote{The Laplacian matrix $L$ is defined to be $D - A$ where $D$ and $A$ denote the degree matrix and adjacency matrix respectively.} of a graph then apply the following proposition [107].

**Proposition 4.** A graph $G$ has $m$ connected components if and only if the algebraic multiplicity of 0 in the Laplacian is $m$. 

53
Algorithm 1

Input:
Data $X = \{x_1, \ldots, x_n\}$; Metric $d(\cdot, \cdot)$; Resolution $r$;

Output:
Influence $\mu(x_1), \ldots, \mu(x_n)$; Entropy $H(D)$;

1: for each pair of data sample $x_i, x_j$ do
2: Calculate the pairwise distance $d(x_i, x_j)$;
3: end for
4: Build the Complex $\Delta$ according to the resolution $r$;
5: for each subcomplex $C \subseteq \Delta$ do
6: Calculate the Laplacian matrix $L$ of $C$;
7: Calculate the multiplicity of eigenvalue 0 of of $L$;
8: end for
9: for $i = 1$ to $n$ do
10: Calculate $s(x_i)$ according to (4.3);
11: end for
12: Set $\mu(x_i)$ as the normalized $s(x_i)$, then $H(X) = -\sum_{i=1}^{n} \mu(x_i) \log \mu(x_i)$;

With the above proposition, we can see that the Betti number $\beta_0(X)$ is equal to the number of zeros in the spectrum of the corresponding Laplacian matrix $L_X$. As such, we propose the Algorithm 1 to calculate the influence score and the entropy under the setting of $k = 0$ and $r$ as some constant. It is evident that our proposed Algorithm 1 always produces a probability measure. A more detailed discussion of the choice of $k$ and $r$ is provided in Section 4.5.

Note that this algorithm is specifically designed for the zeroth dimensional homology group, which is a special case in our Shapley Homology framework. In this case, the metric computed in Algorithm 1 can be interpreted as the betweenness centrality [108] of a graph. However, it is important to note that betweenness centrality does not have a natural correspondence to any topological features. On the contrary, the metric derived for the zero-dimensional homology group precisely matches the Betti number. Furthermore, while it provides conceptual convenience in relating our designed metric with betweenness centrality, this connection should not dim the generality of our framework to other homology groups with higher dimensions.

Here we provide a simple time complexity analysis for our algorithm and propose several possible solutions to accelerate the algorithm for future work. Our algorithm can be decom-
posed into three parts: (1) complex construction; (2) calculation of the graph spectrum; (3) assignment of the Shapley value. For the first step, we need to calculate pairwise distance. From the adjacent matrix, it is clear that the complexity is $O(n^2)$. Second, we need to compute the spectrum of all subcomplexes. As such, in total, the complexity is $O(n^32^n)$, where the complexity of each Laplacian decomposition is $O(n^3)$. For the third step, we sum all the marginal utility for one sample, which results in a complexity of $O(2^n)$. Therefore, the complexity of computing the influence scores for all samples is $O(n2^n)$. Based on the above analysis, we obtain the overall complexity of our algorithm as $O(n^32^n)$. Clearly, the second and third steps contribute most to the complexity. In order to alleviate the computational burden caused by the second step, future work will consider various approaches (e.g., [109]) that approximates the spectrum of a graph. For the third step, several existing approximation algorithms, e.g., C-Shapley and L-Shapley [95], can be considered for approximating the Shapley value using local topological properties.

In the following, we provide a set of case studies on several different graphs with representative topological structures. In particular, we study four types of graphs representing the space of binary strings generated by four regular grammars. We select these grammars due to their simplicity for demonstration. A brief introduction of the selected grammars is provided in Table 4.1. Since we deal with binary string, we specify the distance metric used in these studies as the edit distance [10] and set the radius $r = 1$. Also, we set the length $N$ of generated strings to fixed values as specified in Table 4.1.

Furthermore, we generalize our proposed framework to another six special sets of graphs and provide the analytical results of their vertices’ influence and the entropy values. These graphs are selected as they represent a set of simple complexes that can be used for building up more complicated topological structures.

**Simple Examples on Regular Grammars** Here, we calculate and analyze the first three grammars shown in Table 4.1 as they have simple yet different topological features. As we can see from Table 4.1, for the first grammar, given a fixed $N$, there exists only one string defined by this grammar. As a result, the influence of this sample is 1, and the entropy $H(g_1) = 0$. For the second grammar, it is clear that any pair of valid strings has an edit distance larger than 1. In this case, the complex formed with the radius $r = 1$ consists of

---

2Grammar 1, 2, and 4 are selected from the set of Tomita grammars [110], of which their indices in the Tomita grammars are 1, 5, and 3, respectively.
Table 4.1  Example grammars and their associated entropy values.

<table>
<thead>
<tr>
<th>$g$</th>
<th>Description</th>
<th>Entropy ($\beta_0$)</th>
<th>Entropy ($\beta_1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1^*$</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>even number of 0s and even number of 1s.</td>
<td>log $M$</td>
<td>$\approx \log M$</td>
</tr>
<tr>
<td>3</td>
<td>$1^* + 0^r(1 + 0)$</td>
<td>$3\log 2/2$</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>an odd number of consecutive 1s is</td>
<td>$\approx 2.30$</td>
<td>$\approx 2.12$</td>
</tr>
<tr>
<td></td>
<td>always followed by an even number of</td>
<td>($N = 4$)</td>
<td>($N = 4$)</td>
</tr>
<tr>
<td></td>
<td>consecutive 0s.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

disjoint points. Assuming that there exist $M$ valid strings defined by this grammar with the length of $N$, then all strings have the same influence $1/M$ and the entropy $H(g_2) = \log M$.

For the third grammar, when the length $N$ of its associated strings is larger than 3, then the set $g_3$ of these strings can be expressed as $g_3 = \{1^N, 0^N, 0^{N-1}\}$, denoted as $g_3 = \{1, 2, 3\}$ for notation simplicity. We depict the complex for the case when $r = 1$ in Figure 4.2. According to Proposition 4, we then have the following Betti number $\beta_0$ of each subcomplex.

$$
\beta_0(G_1) = \beta_0(G_2) = \beta_0(G_3) = \beta_0(G_{2,3}) = 1,
\beta_0(G_{1,2}) = \beta_0(G_{1,3}) = \beta_0(G_{1,2,3}) = 2.$$

where $G_S (S \subseteq \{1, 2, 3\})$ denotes the subgraph of $G_{1,2,3}$ formed by the vertex set $S$. According to (4.3) and Algorithm 1, we have $\mu(1) = 0.5, \mu(2) = \mu(3) = 0.25$, and finally the entropy is $H(g_3) = \frac{3}{2} \log 2$.

A Complicated Example  The fourth grammar $g_4$ shown in Table 4.1 is more complicated than the three grammars mentioned above. In particular, let $N = 4$, $r = 1$ or 2, then we illustrate the results in Figure 4.3, and the entropy is $H(g_4) = 2.292$ when $r = 1$ and $H(g_4) = 2.302$ when $r = 2$. Besides the analytical results presented here, we further demonstrate the difference between this grammar and the first two grammars in a set of empirical studies in Section 4.6.2.

A Case Study on Special Graphs In this part, we apply our framework to six special families of graphs, which are shown with examples in Table 4.2. Please see Section 4.8 for a detailed calculation of the influence scores and entropy values for these graphs. With the
Table 4.2 Six special sets of graphs and their values of influence for the vertices based on Betti number $\beta_0$.

<table>
<thead>
<tr>
<th>Example</th>
<th>Shapley Value</th>
<th>Influence Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Graph $K_n$</td>
<td>$1/n$</td>
<td>$1/n$</td>
</tr>
<tr>
<td>Cycles $C_n$</td>
<td>$2/3 - 1/n$</td>
<td>$1/n$</td>
</tr>
<tr>
<td>Wheel Graph $W_n$</td>
<td>Periphery: $1/3 - 1/(n(n - 1))$ Center: $(n^2 - 7n + 18)/6n$</td>
<td>Periphery: $2(n^2 - 3n + 4)/(3(n - 1)(n^2 - 2n + 4))$ Center: $(n^2 - 7n + 18)/(3(n^2 - 3n + 4))$</td>
</tr>
<tr>
<td>Star $S_{n-1}$</td>
<td>Periphery: $1/2$ Center: $(n^2 - 3n + 4)/2n$</td>
<td>Periphery: $n/2(n^2 - 2n + 2)$ Center: $n^2 - 3n + 4/2(n^2 - 2n + 2)$</td>
</tr>
<tr>
<td>Path Graph $P_n$</td>
<td>Ends: $1/2$ Middle: $2/3$</td>
<td>Ends: $3/(2n - 1)$ Middle: $2/(2n - 1)$</td>
</tr>
<tr>
<td>Complete Bipartite Graph $K_{m,n}$</td>
<td>$m$ side: $\frac{n(n-1)}{m(m+1)} + \frac{1}{n+1}$ $m$ side: $\frac{m+n^2+n^2+n+m+n+m+n-m-n}{m(m+n)}$ $m$ side: $\frac{m+n^2+n^2+n+m+n+m+n-m-n}{m(m+n)}$ $m$ side: $\frac{m+n^2+n^2+n+m+n+m+n-m-n}{m(m+n)}$</td>
<td></td>
</tr>
</tbody>
</table>

analytical results shown in Table 4.2, it is easy to derive the following two corollaries:

**Corollary 3.** Based on Betti number $\beta_0$, we have $H(K_n) = H(C_n) > H(W_n) > H(S_{n-1})$ for $n > 5$, where $K_n$ denotes the complete graph, $C_n$ denotes the cycle graph, $W_n$ denotes
Figure 4.3  The Čech complex, influence scores and entropy of the fourth grammar with $N = 4$ when applying Betti number $\beta_0$.

**wheel graph** and $S_n$ denotes the **path graph**.

**Corollary 4.** Suppose that $m + n$ is constant, then based on Betti number $\beta_0$, the value of $H(K_{m,n})$ decreases as $|m - n|$ increases, where $K_{m,n}$ denotes the complete bipartite graph.

**4.4.2 Algorithm and case study of $\beta_1$**

In general, it is extremely challenging to provide an algorithm to compute the Betti number $\beta_1$ of a complex. However, we apply 2-dimensional projection to approximate the homology group. More specifically, we project the complex to a 2-dimensional plane and consider it as a graph. Then it is feasible to calculate the Betti number $\beta_1$ although this approach ignores some high-dimensional information. Hence we have the following proposition [111]:

**Proposition 5.** Let $G$ be a graph with $v$ vertices, $e$ edges, and $m$ connected components, then $\beta_1(G) = v + m - e$.

Note that this is the Euler characteristic of a graph. Similarly, if we want to compute the Betti number $\beta_2$ of a given manifold, we can still project it on a 3-dimensional space and calculate the Euler characteristic of the projected complex. For more information about Euler characteristic and characteristic classes, please refer to [112]. With the above proposition, we propose the Algorithm 2 to calculate the influence score and the entropy under the setting of $k = 1$ and $r$ as some constant.

58
**Algorithm 2**

**Input:**
- Data $X = \{x_1, \ldots, x_n\}$; Metric $d(\cdot, \cdot)$; Resolution $r$;

**Output:**
- Influence $\mu(x_1), \ldots, \mu(x_n)$; Entropy $H(D)$;

1. for each pair of data sample $x_i, x_j$ do
2. Calculate the pairwise distance $d(x_i, x_j)$;
3. end for
4. Build the Complex $\Delta$ according to the resolution $r$;
5. for each subcomplex $C \subseteq \Delta$ do
6. Calculate the number of vertices and edges of $C$;
7. Calculate the Laplacian matrix $L$ of $C$;
8. Calculate the multiplicity of eigenvalue 0 of of $L$;
9. Calculate the Betti number according to Proposition 5;
10. end for
11. for $i = 1$ to $n$ do
12. Calculate $s(x_i)$ according to (4.3);
13. end for
14. Set $\mu(x_i)$ as the normalized $s(x_i)$, then $H(X) = -\sum_{i=1}^{n} \mu(x_i) \log \mu(x_i)$;

From the description of the algorithm, it is clear that the calculation of Betti number $\beta_1$ is based on $\beta_0$, and the time complexity is still dominated by Laplacian decomposition. hence the complexity of Algorithm 2 is still $O(n^32^n)$. However, different from Algorithm 1, there is no guarantee that Algorithm 2 always produces a probability measure. Note that this Shapley Homology framework will give a probability measure if and only if there exists a subcomplex such that the corresponding homology is not equal to zero. In other words, the necessary and sufficient condition to compute $\beta_1$ by Algorithm 2 is that there exists a loop in the projected graph. Hence we can see from Table 4.1 that the entropy of simple grammars is zero. We will provide a detailed discussion on the influence of $k$ in Section 4.5. For the fourth grammar $g_4$, we present the results of sample influence and entropy in Figure 4.4 and Table 4.1, respectively.

**A Case Study on Special Graphs** In this part, we provide the analytical analysis on the sample influence of four special families of graphs based on Betti number $\beta_1$, which are shown with examples in Table 4.3. We give a detailed calculation of sample influence in Section 4.8. Note that the sample influence of path graph and star does not make sense here
since they do not have a loop structure. For a complete bipartite graph, we only keep the $O(m+n)$ term as the approximation of the Shapley value, and the influence score is omitted since the expression is too long. One can easily get the influence score by $l_1$ normalization of the Shapley value. Similar to the corollary 3 and 4, we have the corollaries for Algorithm 2 as well:

Corollary 5. Based on Betti number $\beta_1$, we have $H(K_n) = H(C_n) > H(W_n)$ for $n > 4$, where $K_n$ denotes the complete graph, $C_n$ denotes the cycle graph, $W_n$ denotes wheel graph.
Corollary 6. Suppose that \( m + n \) is constant, then based on Betti number \( \beta_1 \), the value of \( H(K_{m,n}) \) decreases as \( |m - n| \) increases, where \( K_{m,n} \) denotes the complete bipartite graph.

4.5 Discussion

In this section, we will provide a more in-depth discussion on Shapley homology. As previously introduced in Section 4.3.3, our proposed influence score can be viewed as a function of the parameter \( k \) and \( r \). Here we extend our study from the previous cases of \( k = 0 \) and \( k = 1 \) to cover other cases and show that adopting the 0\(^{th}\)-dimensional and 1\(^{st}\)-dimensional homology offer the merits of having more meaningful and practical implication. Also, we discuss the effect of selecting different radii \( r \) on the topological features we study. More importantly, we show that our framework provides a more general view of sample influence and can be further adapted to capture the dynamical features of a topological space. Furthermore, we compare different definitions of entropy and point out the advantages of our proposed entropy. At last, we provide some discussion on the VC dimension and other topological features as well.

4.5.1 Topological Perspective of Shapley Homology

Discussion of the Choice of \( k \). We start from the case \( k = 1 \). In this case, the Betti number \( \beta_1 \) represents the number of 1-dimensional holes in a given manifold. Take the third grammar introduced in Section 4.4 as an example. It is clear to see that when using the 1-dimensional homology, all subcomplex has \( \beta_1 \) equal to 0. This is precisely the case when our Algorithm 2 cannot generate a probability measure. In a more general case, it is easy to show by the homology theory [113] that once all subcomplex gets Betti number \( \beta_{\hat{K}} = 0 \) for some \( \hat{K} \), then for any other \( k \)-dimensional homology with \( k > \hat{K} \), our framework cannot produce a probability measure. When adopting \( k \)-dimensional homology with \( k \) larger than 2, our framework will highlight the effect of having the voids constructed or destroyed in the manifold by introducing a data point. However, the voids in a high-dimensional manifold can be very abstract and cause difficulty for intuitive understanding.

Another difficulty of adopting homology with higher dimensions is due to practical concerns. More specifically, since it is challenging to calculate the homology when \( k \) is large, one may need to apply tools such as the Euler characteristic or Mayer-Vietoris Sequences [103]. Specifically, if the homology of the space consists of finitely generated projective modules,
then the Euler characteristic is the alternating sum of its Betti numbers. In general, we can consider the Euler characteristic as a topological feature and decompose it to each individual point. In this way, taking Betti number $\beta_0$ and $\beta_1$ can be considered as an approximation (analog to the leading terms in Taylor expansion). The idea of Mayer-Vietoris Sequences is to decompose a given topological space $X$ into subspaces $A$ and $B$, whose homology groups, including the homology of the intersection $A \cap B$, may be easier to compute. The Mayer-Vietoris theorem claims that there is a long exact sequence that relates the homology groups of $X$ to the homology groups of $A$, $B$, and $A \cap B$. With the long exact sequence and the connecting morphism, we can compute the higher homology groups. Finally, more advanced theories exist to calculate homology groups, e.g., spectral sequence [114], which is out of the scope of the discussion.

**Discussion of the Choice of $r$.** The radius $r$ plays a critical role in building a manifold. For example, when $r$ is sufficiently small, the resulting complex contains only discrete data points with equal influence in our framework. When $r$ is sufficiently large, on the other hand, the complex is contractible, which indicates that there is no difference between the influence of data points contained in this complex. Both aforementioned extreme cases produce equal importance for the data points in a data set, and correspond to the i.i.d. assumption of sample distribution that is commonly assumed in current machine learning practice. In this sense, our framework provides a general abstraction of sample influence.

In practice, selecting a proper $r$ can be very difficult. This explains the usage of persistent homology [106], which studies the evolution of topological features of space as $r$ dynamically changes. It is not difficult to extend our framework to dynamically generate a series of probability measures for the influence and the corresponding entropy values for a data set. However, a sequence of probability measures with various radii has an unclear physical meaning. In this way, the radius should depend on the specific application and should be pre-defined as the resolution of the problem of interest.

**Stability Analysis** To test whether an algorithm is robust to noise is crucial for real problems. From a more formal perspective, when noise exists, a traditional approach is to leverage stability analysis [115,116]. Specifically, the robustness against noise can be ensured by Lipschitz continuity. A concrete example is the adversarial example problem, which represents a less robust case where a small perturbation in the input space can lead to a large
variation in the output space. Typically, a soft stability theorem ensures that the Lipschitz continuity from the data in an algebraic module is implemented in a topological data analysis. Then for the algebraic module to be influenced, the Shapley value is implemented, which is also computed by Lipschitz continuous functions. As such, we intuitively assume that the overall algorithm should be robust to noise, as it ensures Lipschitz continuity.

Other Groups Associated to a Topological Space Homotopy, homology, and cohomology groups are commonly studied in algebraic topology. Here we briefly discuss the connections and differences between them under our proposed framework. We start from the homotopy group first. The simplest homotopy group is the fundamental group, formed by the sets of equivalence classes of the set of all loops. Intuitively, higher homotopy groups record information about the basic shape, or holes, of a topological space, which are the natural higher-dimensional analogs of the fundamental group [103]. Like homology, a fibration, a continuous mapping satisfying the homotopy lifting property for any space, will induce a long exact sequence in the homotopy groups, which is crucial to homotopy computation. Besides, Hurewicz theorems [117] provides a critical link between homotopy groups and homology groups. Formally, for any path-connected space, there exists a group homomorphism from the $k^{th}$ homotopy group to the $k^{th}$ homology group with integer coefficients. Furthermore, as we consider the case $k = 1$, the Hurewicz homomorphism induces an isomorphism between the abelianization of the fundamental group and the first homology group. However, the number of generators of the fundamental group is not necessarily the same as the Betti number $\beta_1$. For example, consider a knot $K$ and its knot group given by $\pi_1(\mathbb{R}^3 \setminus K)$, the Betti number of this space is 1 but the number of generators of the fundamental group is determined by the knot structure.

On the other hand, cohomology is an algebraic variant of homology, the result of a simple dualization in the definition. Hence, it satisfies similar axioms except that induced homomorphisms go in the opposite direction [103]. Homology and cohomology group of a topological space can be determined each other by duality theorems, e.g., Poincaré duality and Alexander duality. The advantage of cohomology is that it provides an additional internal product structure called a cup product, which furnishes it into a graded ring. Note that we still have a long exact sequence in the cohomological version to compute the higher groups similar to homology. As we used the Betti number in the proposed Shapley homology framework, it is natural to consider using the rank of cohomology group instead. However,
we have the same results by universal coefficient theorem [117], which implies that there is no difference between the torsion-free part of cohomology and homology groups with integer coefficients.

### 4.5.2 Remarks on Related Definitions of Entropy

Different notions of entropy have been previously proposed. Here we briefly revisit several representative definitions and compare them with our entropy definition, which quantifies the number of boundaries in a data set. As such, it determines the capacity of a neural network that is needed to learn this data set [118].

**Graph Entropy** An important property of the graph entropy [119] is monotonicity since it is defined based on mutual information. Specifically, it describes that the entropy of a subgraph is smaller than that of the whole graph on the same vertex set. In our case, by considering the complex as a graph, our entropy is defined to capture the geometric properties (such as the symmetry invariant property mentioned in Section 4.3.3) of a graph. More specifically, our entropy measure focuses more on the variation of the topological features when a graph is changed. As such, our definition also covers variations that may violate the monotonicity property and the subadditivity property.

**Entropy for Grammatical Learning** The entropy is introduced in Chapter 3, which is defined to reflect the balance between the population of strings accepted and rejected by a particular grammar. It shows that the entropy of a certain grammar is equal to that of the complement of that grammar. This raises a contradiction in the intuition that a grammar with a higher cardinality of strings is more likely to have a higher entropy value. Our entropy, on the other hand, is defined to capture the intrinsic properties of a set of samples instead of reflecting the difference between different sets of samples. In this sense, our entropy is more likely to assign a higher entropy value to a set of samples with larger cardinality.

**Entropy in Symbolic Dynamics** This type of entropy [120] is defined to reflect the cardinality of a shift space, which can be regarded as a more general definition of regular grammar. A grammar with high cardinality will have larger entropy, which makes sense in general. It implicitly assumes that any shift contained in a shift space has equal influence. This is contrary to our case in that we define the entropy to describe the complexity of a
topological space that contains vertices with different influence, e.g., \( H(S_{n+1}) \) is smaller than \( H(K_n) \). As such, our entropy provides a more fine-grained description of a topological space.

### 4.5.3 Other Extended Discussion

**Connection to the VC Dimension** Here we provide a preliminary discussion on the connection between our proposed Shapley Homology and Vapnik-Chervonenkis (VC) dimension [121]. Recall this essentially reflects the complexity of a space of functions by measuring the cardinality of the largest set of samples that can be “shattered” by functions in this space. From a similar point of view, we expect the topology of data space is also critical in evaluating the complexity of real-world learning algorithms in learning a given set of data.

Note that the proposed approach has a close connection to statistical learning theory. An obvious interpretation of our complexity is that it can be viewed as analogous to sample complexity [122], which is closely related to the VC dimension. However, here we point out the limitation of complexity specified by the VC dimension, which is part of the motivation of this work. Specifically, given a certain data space, only the hypothesis space of models with sufficiently large VC dimensions can shatter this data space. For this hypothesis space, we can further use sample complexity to specify the number of samples required to achieve certain PAC learning criteria. However, we argue that different shattering of the data space leads to different levels of complexity (or different entropy values in our terms). Instead of focusing only on the maximally shattered data space, we argue that in practice, when building a learning model, a different shattering should be treated differently. To better explain this effect, we take regular grammars as an example case. One can consider a certain binary regular grammar as a certain configuration in the hypothesis space. In other words, a binary regular grammar explicitly split all the \( 2^N \) strings into the set of accept strings and the set of rejected strings, given that the strings have a fixed length of \( N \). Since this grammar is equivalent to a DFA, and if we regard this DFA as a classification model, it itself has a certain VC dimension [123]. Indeed, this effect is shown in the experiments in Section 4.6 on grammar learning. In particular, in our experiments, the demonstrated different levels of difficulty of learning different regular grammars indicate that different grammars (or different ways of shattering the data space) should not be taken as equal.
Topological Perspective of Regular Grammar Classification  Recall that we categorized regular grammars into three different subclasses and provided several classification theorems based on different representations in Chapter 3. Similarly, motivated by the idea of Shapley homology, a topological perspective of regular grammar classification would be interesting as well. Again, we apply edit distance to topologize the data space\textsuperscript{3} and consider the minimal dimension of Euclidean space it can be embedded into. This work \cite{124} showed that given the mutual distance matrix of data points, we can construct another matrix whose rank determines the dimension of the smallest Euclidean space containing such points. Specifically, it proposed the following lemma:

**Lemma 3.** Given a set of \( n \) points, and let \( d_{ij} \) denotes the mutual distance of point \( i \) and \( j \), then the dimensionality of this set of points is equal to the rank of \( (n-1) \times (n-1) \) matrix \( B \), where \( b_{ij} = (d_{in}^2 + d_{jn}^2 - d_{ij}^2)/2 \).

With the Lemma 3, we can efficiently verify that Tomita grammar 1 lies in \( \mathbb{R}^1 \). For those complicated grammars, we can only truncate the grammar by length to approximate the dimensionality since the dimension of the matrix \( B \) will be infinite.

### 4.6 Experiments

In this section, we first demonstrate the results of using our Algorithms 1 and 2 to identify influential nodes in random graphs. Then we evaluate this on several data sets generated by regular grammars to determine if data sets assigned by our algorithms with higher entropy values cause more challenges for neural networks to be able to learn these grammars. The settings of all parameters in the experiments are provided in Table 4.4.

#### 4.6.1 Graph Classification

For these experiments, we first constructed the learning models and data sets of random graphs following \cite{125}. We adopted this similar setting for the purpose of evaluating and verifying the influence of individual nodes in a graph. To avoid the evaluation results from

---

\textsuperscript{3}Another approach to topologize regular grammar is given by the distance for which words are close if they coincide on a long interval centered at 0 \cite{2}. Formally speaking, the distance is defined as \( d(x, y) = 2^{-r(x,y)} \), where \( r(x,y) = \max\{n \geq 0 \mid x_i = y_i, -n \leq i \leq n\} \). This definition will result in excellent properties in terms of algebraic structure and symbolic dynamics; however, it fails to provide a clear geometric intuition.
being biased by a particular synthetic data set, we performed 20 experimental trials by varying the probability used in the Erdos-Renyi random graph model to generate the same number of random graphs. More specifically, in each trial, we generated a set of 6,000 undirected random graphs from three classes (2000 for each class), and split the generated set into a training and a testing set with the ratio of 9 to 1.

We constructed the learning models based on structure2vec [126] for the task of determining the number of connected components (up to 3) and the Euler characteristics (up to 3) of a given graph. The constructed models function as verification tools for the fidelity test. The idea is to first mask nodes in a graph assigned by our algorithms with different influence scores and examine how much the classification results for this graph are affected. Similar fidelity tests have been widely adopted in previous research on studying feature influence [95,96,127]. In each trial, we first trained a neural network on the training set with random initialization. For each graph in the testing set, we computed the influence score for each node in this graph. Then we generated two data sets, $D_{\text{top}}$ and $D_{\text{bottom}}$ with graphs masked out the top-$J$ nodes and bottom-$J$ nodes (both the top-$J$ nodes and bottom-$J$ nodes are identified by their influence scores). We show in Figure 4.5 the examples of a clean graph and its top-$1$ and bottom-$1$ masked versions. We also constructed a third data set $D_{\text{rand}}$ by randomly masking $L$ nodes for all testing graphs. The evaluation results were obtained by

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Graph</th>
<th>Model</th>
<th>Grammar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Hidden</td>
<td>Model</td>
<td>Hidden (Total)</td>
</tr>
<tr>
<td>Struct2vec</td>
<td>32</td>
<td>SRN</td>
<td>200 (40800)</td>
</tr>
<tr>
<td>Neural Network</td>
<td>16</td>
<td>GRU</td>
<td>80 (19920)</td>
</tr>
<tr>
<td>LSTM</td>
<td></td>
<td>LSTM</td>
<td>80 (19920)</td>
</tr>
<tr>
<td>Optimizer</td>
<td></td>
<td>Rmsprop</td>
<td></td>
</tr>
<tr>
<td>Dataset Setting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>3</td>
<td>Class</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>6000</td>
<td>Total</td>
<td>G1 8677 G2 8188 G4 8188</td>
</tr>
<tr>
<td>Training (per class)</td>
<td>5400 (1800)</td>
<td>Training</td>
<td>6073 5730 5730</td>
</tr>
<tr>
<td>Testing (per class)</td>
<td>600 (200)</td>
<td>Testing</td>
<td>2604 2458 2458</td>
</tr>
<tr>
<td>Prob.</td>
<td>0.02 - 0.21</td>
<td>Length</td>
<td>2- 13</td>
</tr>
<tr>
<td>Nodes</td>
<td>8-14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(a) The original graph  (b) Graph manipulated by masking the most influential node.  (c) Graph manipulated by masking the least influential node.

**Figure 4.5** Example graphs with classification results for each graph from left to right: 1, 2 and 1. This indicates the number of connected components recognized by a neural network on these graphs.

comparing the classification performance of our trained models on these three data sets and that achieved on the original testing sets.

**Figure 4.6** Accuracy of neural networks obtained on data sets with varying scales of manipulations.

Results in Figure 4.6 show all trials that fit the shaded area for each line plot. The accuracy value indexed by $L = 0$ is the averaged accuracy of our models obtained on the clean testing sets from all trials. It is clear from Figure 4.6a and 4.6b that the influence
score calculated by our algorithms effectively indicates the impact of a node on determining the connectivity and Euler characteristics of the graph containing this node, respectively. In addition, as we increase $L$, the accuracy of our models obtained on $D_{\text{top}}$, $D_{\text{bottom}}$ and $D_{\text{rand}}$ degrades with different scales. In particular, the accuracy obtained on $D_{\text{top}}$ and $D_{\text{bottom}}$ shows the largest and smallest scales of degradation, respectively. The result for $D_{\text{top}}$ is surprising in that even on these simple synthetic graphs, the robustness of a neural network model is far from satisfactory. Specifically, similar to the results shown by [125], by masking top-$1$ influential nodes in the testing graphs, the accuracy of a neural network is brought down to $40\% \sim 50\%$.

In the case when we only focus on the vertex with the highest importance score, the experiment can be taken as a sanity check. However, as shown in Figure 4.6, when we evaluate the influence from manipulating vertices that have been identified to have different levels of influence on the performance of classifying graphs, our framework also provides a quantitative evaluation of the importance of different vertices, which goes beyond a sanity check.

### 4.6.2 Grammar Recognition

In this set of experiments, we used three Tomita grammars introduced in Table 4.1 due to their simplicity and wide adoption in various grammar learning research [24, 33, 45]. In Table 4.5, we show for each grammar its entropy value with different radii calculated by our algorithms. We set the radius as discrete integers since we apply the edit distance to regular grammars. Note that it can be computationally intractable if we apply the Shapley homology framework to a large-scale data set since our algorithm under our framework had exponential time complexity. Therefore, we apply an approximation, i.e., we compute the spectrum of $K$ subcomplexes instead of all subcomplexes, where $K$ is a fixed number. More specifically, when computing the specific sample influence, we randomly selected $K$ subcomplexes that contain this sample and use the same formula to obtain its Shapley value.

We used these grammars to generate three sets of binary strings with lengths ranging from 2 to 13 and split the data set of each grammar into a training set and testing set with the ratio of 7 to 3. We then trained several different recurrent networks (Simple Recurrent Network (SRN) [26], gated-recurrent-unit networks (GRU) [30] and long-short-term-memory networks (LSTM) [29]) for a binary classification task on the data set generated by each
grammar. Essentially, the binary classification task is designed to train a recurrent network to correctly differentiate and accept strings that are accepted by a certain regular grammar and reject strings that are rejected by this grammar. For each type of recurrent network, we use the same number of parameters for models used for all three grammars in order to avoid any bias that these models would have any different learning ability. We trained the model of each type of RNN on each grammar for 20 trials. In each trial, we randomly split the training and testing set and randomly initialize the model. The results are demonstrated in Figure 4.7, in which the results from all trials fit the shaded area associated with each plot.

In Figure 4.7a, 4.7b and 4.7c, we see that for the first and fourth grammars, which have lower entropy values, the learning process converges much faster and more consistently than
Table 4.5  Entropy of different grammars with various radii ($K = 100$).

<table>
<thead>
<tr>
<th></th>
<th>Grammar 1</th>
<th>Grammar 2</th>
<th>Grammar 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entropy on $\beta_0$</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r = 1$</td>
<td>2.38</td>
<td>5.67</td>
<td>5.07</td>
</tr>
<tr>
<td>$r = 2$</td>
<td>2.47</td>
<td>5.65</td>
<td>5.08</td>
</tr>
<tr>
<td>$r = 3$</td>
<td>2.46</td>
<td>5.65</td>
<td>5.07</td>
</tr>
<tr>
<td><strong>Entropy on $\beta_1$</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r = 1$</td>
<td>—</td>
<td>6.35</td>
<td>5.21</td>
</tr>
<tr>
<td>$r = 2$</td>
<td>2.76</td>
<td>6.38</td>
<td>5.20</td>
</tr>
<tr>
<td>$r = 3$</td>
<td>2.54</td>
<td>6.51</td>
<td>5.12</td>
</tr>
</tbody>
</table>

The process for the second grammar, which has the highest entropy value. This effect holds for all types of recurrent networks we evaluate. To better illustrate the difference of the difficulty of learning the first and fourth grammars, we provide in Figure 4.7d, 4.7e and 4.7f the zoomed view for each plot at the top row of Figure 4.7. While the learning process of all models converges within ten epochs for both grammars, it is still clear that the learning process is slower for the fourth grammar. These results agree with both our analysis on the entropy of these grammars and the intuition. Specifically, the second grammar defines two sets of strings with equal cardinality when the string length is even. In this case, by flipping any binary digit of a string to its opposite (e.g., flipping a 0 to 1 or vice versa), a valid or invalid string can be converted into a string with the opposite label. This implies that a model must pay equal attention to any string to learn the underlying grammar. This corresponds to our analysis result that in the data space defined by the second grammar, each string sample shares equal influence on affecting the topological feature of this space.

In order to evaluate the sample influence in a more convincing scenario, we inject noise into the training sets of Grammar 4 by randomly selecting several training samples and flipping their labels. To avoid causing an RNN to be severely biased by this noise, we carefully limit the number of samples selected to 5, i.e., picking 5 positive strings and changing their label as negative. Similar to the graph classification experiment, we first calculate the sample influence and select top-5, bottom-5, and random-5 to flip. We then train an RNN on the noisy training set and compare the performance. The evaluation result is shown in Table 4.6, and we found out that all RNN models have a similar level of noise tolerance regardless of which Betti number is applied. More specifically, the tendency of sample influence based on Betti number $\beta_0$ is more obvious, and LSTM/GRU are more likely to overfit. The accuracy of flipping top-5 is higher than flipping random-5 and bottom-5, which indicates that for
Table 4.6  Accuracy of different RNNs obtained on data sets with noise. (The number in bracket denotes the number of misclassified samples.)

<table>
<thead>
<tr>
<th></th>
<th>No Flip</th>
<th>Bottom-5</th>
<th>Random-5</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy on $\beta_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRN</td>
<td>1.0 (0)</td>
<td>0.999 (2)</td>
<td>0.999 (3)</td>
<td>0.998 (5)</td>
</tr>
<tr>
<td>LSTM</td>
<td>1.0 (0)</td>
<td>0.998 (4)</td>
<td>0.998 (4)</td>
<td>0.996 (10)</td>
</tr>
<tr>
<td>GRU</td>
<td>1.0 (0)</td>
<td>0.998 (4)</td>
<td>0.998 (5)</td>
<td>0.996 (9)</td>
</tr>
<tr>
<td>Accuracy on $\beta_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRN</td>
<td>1.0 (0)</td>
<td>0.999 (2)</td>
<td>0.999 (2)</td>
<td>0.998 (4)</td>
</tr>
<tr>
<td>LSTM</td>
<td>1.0 (0)</td>
<td>0.998 (4)</td>
<td>0.998 (4)</td>
<td>0.997 (7)</td>
</tr>
<tr>
<td>GRU</td>
<td>1.0 (0)</td>
<td>0.998 (4)</td>
<td>0.998 (4)</td>
<td>0.998 (6)</td>
</tr>
</tbody>
</table>

bottom-5, the model can easily capture the pattern of the grammar, while for top-5 the model fails. This validates our assumption that different samples have a different level of influence on the learning task.

4.7 Conclusion

We proposed a Shapley Homology framework to study the sample influence on topological features of data space and its associated entropy. This provides an understanding of the intrinsic properties of both individual samples and the entire data set. We also designed two algorithms based on Betti number $\beta_0$ and $\beta_1$ for decomposing the topological features using a cooperative game theory and provide analytical results for several special representative topological spaces. Several advanced topics, including the VC dimension and algebraic topology, are discussed in this chapter. Furthermore, we empirically verified our results with two carefully designed experiments. We show that data points identified by our algorithms that have a more considerable influence on the topological features of their underlying space also have more impact on the accuracy of neural networks for determining the connectivity of their underlying graphs. We also show that regular grammar with higher entropy has more difficulty of being learned by neural networks. In the next chapter, we will have a deeper understanding of the proposed Shapley homology framework as a functor, which facilitates us to build a category-based framework to investigate the mechanism of grammar transfer learning.
4.8 Appendix: Calculation of the Results for the Special Sets of Graphs

Here we provide calculation for results shown in Table 4.2 and 4.3. Only the calculation of Shapley value is given, and influence score can be easily obtained by normalization of Shapley value. With these results, one can easily derive corollaries using basic inequalities.

4.8.1 Results of Betti number $\beta_0$

**The complete graph and cycles** For the complete graph and cycles, the results presented in Table 4.2 are trivial since these two graphs preserve invariance under permutation.

**The wheel graph** For wheel graph shown in Table 4.2, we have

\[
\begin{align*}
\text{Periphery:} & \quad \frac{1}{N!} \left( \sum_{k=0}^{N-4} \binom{N-4}{k} k!(N-k-1)! + \sum_{k=0}^{N-5} \binom{N-4}{k} (k+2)!(N-k-3)!, \right) \\
\text{Center:} & \quad \frac{1}{N} + \frac{1}{N!} \sum_{m=2}^{N-3} \sum_{k=2}^{l} T(N-1, k, m)(k-1)m!(N-m-1),
\end{align*}
\]

where $l = \min(m, N - m - 1)$ and $T(N, k, m) = \binom{m}{k} \binom{N-m-1}{k-1}$, and simplifying the above formulas gives the results in Table 4.2.

**The star graph** For the star graph, we have

\[
\begin{align*}
\text{Periphery:} & \quad \frac{1}{N!} \sum_{k=0}^{N-2} \binom{N-2}{k} k!(N-k-1)! = \frac{1}{2}, \\
\text{Center:} & \quad \frac{1}{N} + \frac{1}{N!} \sum_{k=2}^{N-1} \binom{N-1}{k} (k-1)(k-1)!(N-k)! = \frac{N^2 - 3N + 4}{2N}.
\end{align*}
\]

**The path graph** For the path graph we have

\[
\begin{align*}
\text{Periphery:} & \quad 1, \\
\text{Center:} & \quad \frac{1}{N} + \frac{1}{N!} \sum_{k=2}^{N-1} \binom{N-1}{k} (k-1)!(N-k)!
\end{align*}
\]
Ends: \[ \frac{1}{N!} \sum_{k=0}^{N-2} \binom{N-2}{k} k!(N-k-1)! = \frac{1}{2}, \]

Middle: \[ \frac{2}{N!} \sum_{k=0}^{N-3} \binom{N-3}{k} k!(N-k-1)! = \frac{2}{3}. \]

**The complete bipartite graph** For complete bipartite graph we have

\[
\text{m side: } \frac{1}{(m+n)!} \sum_{k=0}^{m-1} \binom{m-1}{k} k!(m+n-k-1)! + \sum_{k=2}^{n} \binom{n}{k} (k-1)! (m+n-k-1)! ,
\]

\[
\text{n side: } \frac{1}{(m+n)!} \sum_{k=0}^{n-1} \binom{n-1}{k} k!(m+n-k-1)! + \sum_{k=2}^{m} \binom{m}{k} (k-1)! (m+n-k-1)! ,
\]

which after simplifying the above gives the results in Table 4.2.

**4.8.2 Results of Betti number \( \beta_1 \)**

The calculation of sample influence of complete graph and cycles based on \( \beta_1 \) are exactly the same as \( \beta_0 \). We only focus on the wheel graph and the complete bipartite graph.

**The wheel graph** For wheel graph, there is no distinction between the influence score of the center point and that of complete graph. For the periphery, we have:

\[
\text{Periphery: } \frac{2}{N!} \sum_{k=4}^{N-2} \binom{N-3}{k-3} (k-1)! (N-k)! = \frac{2N^2 - 2N + 3}{3N(N-1)}. 
\]

**The complete bipartite graph** For complete bipartite graph we have:

\[
\text{m side: } \frac{1}{(m+n)!} \sum_{k=4}^{m+n} \min\{m-1,k-2\} \binom{m-1}{t} \binom{n}{k-1-t} (k-t-2)(k-1)! (m+n-k)!
\]

\[
\text{n side: } \frac{1}{(m+n)!} \sum_{k=4}^{m+n} \min\{n-1,k-2\} \binom{n-1}{t} \binom{m}{k-1-t} (k-t-2)(k-1)! (m+n-k)!.
\]

The result can be decomposed into two parts: the dominated part \( D \) and the residual.
part $R$, where

$$D = \frac{m + n - 3}{2} - \frac{(m - 1)(m + n + 2)(m + n - 3)}{2(n + m - 1)(m + n)}.$$ 

$$R = \frac{(m - 1)(m - 2)(m - 3) + 2nm(n - 1)(n - 2)}{(n + 1)(n + m - 1)(m + n)(n + m - 2)} + \frac{n(n + m - 1)(m + n - 2) + 3nm(n - 1)}{m(m + n + 1)(m + n - 1)(n + m - 2)},$$

for $m$ side Shapley value. For $n$ side, simple replacing $m$ with $n$ and $n$ with $m$ will give the results.

The above derivation has applied the following facts which can be easily verified by combinatorics:

$$\frac{1}{N!} \sum_{k=0}^{N-m} \binom{N-m}{k} k!(N-k-1)! = \frac{1}{m},$$

$$\frac{1}{(m+n)!} \sum_{k=2}^{m} \binom{m}{k} (k-1)! (m+n-k-1)! = \frac{m(m-1)}{n(n+1)(m+n)},$$

$$\frac{1}{N!} \sum_{m=2}^{N-3} \sum_{k=2}^{l} T(N-1, k, m)(k-1)m!(N-m-1)! = \frac{(N-3)(N-4)}{6N},$$

where $l = \min(m, N - m - 1)$ and $T(N, k, m) = \frac{N}{m} \binom{m}{k} \binom{N-m-1}{k-1}$. 

75
Chapter 5
Deep Learning, Grammar Transfer, and Transportation Theory

Despite its widespread adoption and success, deep learning-based artificial intelligence is limited in providing an understandable decision-making process of what it does. This makes the “intelligence” part questionable since we expect real artificial intelligence to not only complete a given task but also perform in an understandable way. One way to approach this is to build a connection between artificial intelligence and human intelligence. In this chapter, we use grammar transfer to demonstrate a paradigm that connects these two types of intelligence. Specifically, we define the action of transferring the knowledge learned by a recurrent neural network from one regular grammar to another grammar as grammar transfer. We are motivated by the theory that there is a natural correspondence between second-order recurrent neural networks and deterministic finite automata, which are uniquely associated with regular grammars. To study the process of grammar transfer, we propose a category-based framework we denote as grammar transfer learning. Under this framework, we introduce three isomorphic categories and define ideal transfers by using transportation theory in operations research. Regarding the optimal transfer plan as a sensible operation from a human perspective, we then use it as a reference for examining whether a learning model behaves intelligently when performing the transfer task. Experiments under our framework demonstrate that this learning model can learn grammar intelligently in general but fails to follow the optimal way of learning.
5.1 Introduction

It is well known that deep neural architectures provide good representations for most data by using multiple processing layers that offer multiple levels of abstraction. These models have dramatically improved the state-of-the-art performance in many applications such as computer vision, natural language processing, indoor localization, etc. [11] It is believed that deep learning (DL) based artificial intelligence (AI) can improve performance for many problems in several other domains [128].

However, DL has raised concerns about its innate intelligence. For example, Gary Marcus [129] doubts that DL models can learn causality since they are often built to capture the concurrence of two events from a stateless statistical perspective. Furthermore, to capture statistics, DL usually requires a very large amount of data for training. This aspect of DL brings questions to their innate functionality when the application scenario changes, especially when the captured data changes over time or the amount of data is limited. Maybe because of this, recent research has focused more on model performance and less on model intelligence. This type of ”positivistic puritanism” has been observed in other fields such as physics [130] and should raise similar issues for researchers in DL.

Some ask if machines can even ever have intelligence [131]. While machines readily outperform humans in memorization and calculation, it is not evident these advantages lead to real intelligence but instead offer some narrow idiot-savant skills [132]. It is more likely that these advantages of machines are so dominant that they offer a biased sense of intelligence. Indeed, the lack of reasoning in DL models has already been shown in tasks that involve abstraction, causal reasoning, and generalizing, e.g., self-driving cars, mathematical induction, and healthcare analytics [133–135]. Also, many find it hard to believe that real intelligence can be generated from improved hardware and computational power. Unfortunately, the canonical Turing test, which examines a machine’s ability to exhibit intelligent behaviours that are indistinguishable from those of a human \(^1\), cannot set a practical standard for current research in AI. Instead of evaluating the intelligence of DL models, another approach [137, 138] is to bring some self-consciousness to models by inserting in them distilled knowledge. While it is arguable if external forces can bring

\(^1\)The Turing test offers a standard for intelligence from a linguistic perspective [136]. It is also a practice of linguistic turn in philosophy, a part of analytic philosophy. Analytic philosophy has become the mainstream of contemporary philosophy nowadays, and in a sense, it validates the rationality of the Turing test for intelligence.
intelligence to machines, when the observed results follow the expectation, these successful cases are assumed to generalize without sufficient caution.

Our approach, instead of investigating the intelligence of DL models from a performance-oriented perspective, is to focus on its learning process. Specifically, we embed this approach as a testing method and demonstrate preliminary results obtained by applying our method to the grammar transfer problem. Our test method can be summarized as:

1. **Analogy.** Establish an analogy between machine intelligence and a reference of intelligence, e.g., human intelligence, by applying category theory [139] for building a reference. Specifically, for our built reference we provide two levels of correspondence – object correspondence and morphism correspondence – between the typical training operations in DL methods and an optimal operation obtained by using transportation theory in operations research [140]. Details are discussed in Section 5.3.

2. **Learning Process.** Investigate the learning process of a model by using the setting of grammar transfer. This is because grammar transfer facilitates the building of object correspondence as previously mentioned, and provides straightforward metrics for examining the morphism correspondence.

3. **Theoretical Basis.** Understand the critical limitation of existing methods \(^\text{2}\) for explaining DL models since there is not an established theoretical connection between learned knowledge and human knowledge. In our setting, this problem is mitigated since a regular grammar (or deterministic finite automata (DFA)) has a natural correspondence with second-order recurrent neural networks [144]. Moreover, the learning cost can be measured quantitatively and compared with the transportation cost [140], which we assume to reflect human intelligence.

In summary, we propose a novel framework to investigate the mechanism of grammar transfer learning, with the intent of helping understand the foundation and intelligence of deep learning models.

\(^{2}\)These methods propose to either decompose the output decision of a model into its input, which can have a linear or nonlinear form at the feature or instance level [96,141,142], or employ interpretable models to approximate black box DL models [127,143]
5.2 Preliminary

This section introduces mathematical tools and definitions that are used throughout the chapter and reviews several related works as well.

5.2.1 Mathematical Preliminaries

**Shapley Homology.** Recall that Shapley homology proposed in Chapter 4 provides a quantitative metric for the influence of individual data samples on the topological features of a set of samples. Essentially, it transforms a metric space into a discrete measure space by applying the techniques in algebraic topology and combinatorics. More specifically, the Shapley homology first constructs a simplicial complex by choosing a specific resolution, then decomposes the Betti number of the complex into the Shapley values of individual samples. Finally, the Shapley values of all samples are normalized by the $L_1$ norm to obtain a probability distribution. Here we applied Shapley Homology as a functor between two proposed categories.

**Optimal Transport.** Transportation theory studies optimal transportation and allocation of resources, and was first proposed and formalized by Gaspard Monge in 1781 [145]. The main goal is to perform transportation with the minimum cost. More formally, let $X$ and $Y$ be two metric spaces and $c : X \times Y \to [0, \infty]$ be a measurable function. Given probability measures $\mu$ on $X$ and $\nu$ on $Y$, Monge’s formulation of the transportation problem is to find a transportation map $T : X \to Y$ that realizes the infimum [146]:

$$\inf \left\{ \int_X c(x, T(x))d\mu(x) \mid T_*(\mu) = \nu \right\}, \quad (5.1)$$

where $T_*(\mu)$ denotes the push forward of $\mu$ by $T$. Note that Monge’s formulation can be ill-posed. As such, Kantorovich introduced a new formulation that uses a probability measure $\gamma$ on $X \times Y$ that attains the infimum [146]:

$$\inf \left\{ \int_{X \times Y} c(x, y)d\gamma(x, y) \mid \gamma \in \Gamma(\mu, \nu) \right\}, \quad (5.2)$$

where $\Gamma(\mu, \nu)$ denotes the collection of all probability measures on $X \times Y$ with marginals $\mu$ on $X$ and $\nu$ on $Y$. Furthermore, the p-Wasserstein metric is defined by setting the cost
function as \( c(x, y) = |x - y|^p \):

\[
W_p(\mu, \nu) = \left( \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{X \times Y} |x - y|^p \, d\gamma(x, y) \right)^{1/p}.
\] (5.3)

A more detailed introduction of this subject can be found here [140].

**Category Theory.** Category theory is a general theory of mathematical structures. More specifically, it studies the universal components of a family of structures of a given kind and how structures of different kinds are interrelated in an abstract and conceptual framework [147]. There was early work on tying together category theory and neural networks [148]. Here we introduce some basic concepts that will be used to build up the framework.

**Definition 11 (Category).** A category \( \mathbf{C} \) consists of:

1. A class \( \text{ob}(\mathbf{C}) \), whose elements are called objects;
2. A set \( \text{hom}_\mathbf{C}(a, b) \) for every pair of objects \( a \) and \( b \), whose elements are called morphisms;
3. For every triad of objects \( a, b, \) and \( c \), a function, named as composition, \( \text{hom}_\mathbf{C}(a, b) \times \text{hom}_\mathbf{C}(b, c) \to \text{hom}_\mathbf{C}(a, c) \) have its value at \( (f, g) \) denoted by \( g \circ f \).

They are required to satisfy the following conditions:

1. (Associative) If \( f : a \to b, \ g : b \to c, \) and \( h : c \to d \), then \( h \circ (g \circ f) = (h \circ g) \circ f \);
2. (Identity) For every object \( a \), a distinguished element \( 1_a \in \text{hom}_\mathbf{C}(a, a) \), called identity on \( a \), such that for every morphism \( f : a \to b \) we have \( 1_b \circ f = f = f \circ 1_a \).

We then have the following definition of a functor:

**Definition 12 (Functor).** Let \( \mathbf{C} \) and \( \mathbf{D} \) be categories. A functor \( F \) from \( \mathbf{C} \) to \( \mathbf{D} \), denoted as \( F : \mathbf{C} \to \mathbf{D} \), consists of:

1. for each object \( a \) in \( \mathbf{C} \), an object \( F(a) \) in \( \mathbf{D} \);
2. for each morphism \( f : a \to b \) in \( \mathbf{C} \), a morphism \( F(f) : F(a) \to F(b) \).

that satisfies

1. for every object \( a \) in \( \mathbf{C} \), \( F(1_a) = 1_{F(a)} \);
2. for all morphisms \( f : a \to b \) and \( g : b \to c \), \( F(g \circ f) = F(g) \circ F(f) \).

Finally we say that two categories \( \mathbf{C} \) and \( \mathbf{D} \) are isomorphic if there exist functors \( F : \mathbf{C} \to \mathbf{D} \) and \( G : \mathbf{D} \to \mathbf{C} \) that are mutually inverse to each other.
Transfer Learning. Transfer learning investigates the transfer of knowledge in a learning model, which has been trained with data collected from a source domain to a target domain that is different but related to the source domain. This technique can mitigate problems caused by having insufficient training data [149]. In a more formal way [150], given a source domain $D_S$ and a learning task $T_S$, a target domain $D_T$ and a learning task $T_T$, transfer learning aims to improve the learning of the target predictive function $f_T(\cdot)$ in $D_T$ using the knowledge in $D_S$ and $T_S$.

5.2.2 Related Work

Criticisms and Measurement of Artificial Intelligence. Artificial intelligence has faced many different challenges and criticisms over the past few decades, most of which are from a philosophical perspective [151–153]. The most famous one is Hubert Dreyfus’s views on artificial intelligence [154,155], which presented a pessimistic assessment of AI’s progress and a critique of the philosophical foundations of the field. He also identified and challenged four philosophical assumptions that supported the faith of early AI researchers. With the development of philosophy, people applied more advanced tools to analyze AI, e.g., postmodernism [156] and linguistic philosophy [157]. Recent research showed that most animal behavior is not the result of clever learning algorithms but is encoded in the genome [158], which provided evidence of a critique of pure learning from a biological perspective. In addition, Gary Marcus pointed out several limitations of deep learning [129]. He believed that deep learning is greedy, brittle, opaque, and shallow, and it is not ‘a universal solvent, but one tool among many.’ He also criticized the excessive publicity, incomplete report of AI, and how the AI community conducts their research.

Suggestions on how to improve AI have been sparse, especially on the measurement of artificial intelligence. This work [159] discussed the possibility of devising an intelligence quotient for machines and provided several conjectures. Recent work [160] introduced a new perspective on defining and evaluating intelligence, claiming that a measure of intelligence must account for priors, experience, and generalization difficulty, and therefore all intelligence is relative to the particular scope of application.

Category Theory in Machine Learning. Category theory has been seldom used in machine learning. Nevertheless, this does not detract from the utility. It should be noted
Researchers in the early 2000s realized that category theory could benefit in the understanding of a neural network by applying it to its knowledge and cognition aspects [161, 162]. Recent attempts to connect machine learning with category theory [163] show that a linear simplification of a recursive neural tensor network can be mapped directly onto a categorical approach. Also, a structural perspective on back-propagation [164] was provided in the language of category theory. At the same time, category theory has been widely adopted in cognitive science [165, 166]. This is because analogy, which naturally fits in category theory, is claimed to be at the core of human cognition. Motivated by this, our goal is to build a category-based framework to provide a new view on analogy-related artificial intelligence.

**Optimal Transport and Transfer Learning.** Optimal transport theory was introduced in DL by the Wasserstein generative adversarial network (WGAN) [167]. It has also been used to improve transfer learning and domain adaptation [168–171]. In general, these studies primarily target feed-forward neural networks built for computer vision and natural language processing tasks and append the original loss function with a transport loss [171] or learn a transportation plan that matches the probability distributions of the source and target domains [172]. Unlike these studies, we align the transport loss (measured by the Wasserstein edit distance) with the effort taken to transfer a learning model between different regular grammars.

5.3 Framework

In this section, we will build our framework by applying the preliminaries introduced in Section 5.2.

5.3.1 Grammar Transfer Learning

Different from transfer learning, the main goal here is to examine whether the transferring process of a learning model is aligned with an optimal transportation plan. Specifically, in grammar transfer, we focus on the process of transferring the knowledge learned by 2-RNN between different regular grammars. We use strings accepted by different DFA as the data sets. These choices of data and model are based on the theoretical equivalence between a 2-RNN and DFA. In this way, we are able to build a numerical connection between the metric used in transportation theory and the metric used in grammar transfer learning. In
order to adapt eq.(5.3) for grammar transportation, we define a Wasserstein edit distance as follows:

\[ W_e(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{X \times Y} d_e(x, y) d\gamma(x, y), \] (5.4)

where \( d_e(x, y) \) denotes the edit distance that measures the minimum number of operations needed to convert a string \( x \) into another string \( y \) [10].

5.3.2 Category

It is difficult to keep track of the grammar transfer process by only focusing on the models. Fortunately, category theory provides an alternative perspective. More precisely, we define three isomorphic categories to provide different representations of a regular grammar: a regular grammar category, a probability distribution category, and a learning model category.

We first start with the regular grammar category, denoted as \( \text{RG} \). Here we limit our discussion to a finite number of samples since this is more realistic in practice. Suppose we have \( N \) samples. Then we can take grammar as a definitive separation of positive and negative samples. Let \( \text{ob}(\text{RG}) \) denote the collection of all possible separations and \( \text{hom}(\text{RG}) \) contain all abstract transitions from one separation to another.

Recall that Shapley homology will assign a unique measure to a given grammar with a fixed resolution. Then we can define the probability distribution category as \( \text{PD} \), of which the objects are the probability distributions associated with objects in \( \text{ob}(\text{RG}) \). Similarly, the morphisms contain all the transitions from one distribution to another.

Assume that a 2-RNN is trained with fixed hyper-parameters and random seeds, and the training process is stopped immediately after this 2-RNN successfully learns the underlying grammar (all strings are classified correctly). Then this 2-RNN is uniquely determined by the grammar. Hence, we denote the learning model category as \( \text{ob}(\text{LM}) \), which contains 2-RNNs associated with different grammars and denote the grammar transfer learning as \( \text{hom}(\text{LM}) \). Based on our definitions, the cardinality of \( \text{ob}(\text{RG}) \), \( \text{ob}(\text{PD}) \), and \( \text{ob}(\text{LM}) \) are equal to each other. However, the morphism for each category fails to possess a well-defined correspondence. This will be addressed in the latter part of this section.
5.3.3 Framework

Here, we introduce some necessary constraints on the morphisms to formally describe the optimal plan in our context of grammar transfer. Note that introducing these constraints will invalidate the categories described above since the composition in each category makes no longer makes sense. However, this will not affect our discussion. As such, we keep using the concept in category theory and the corresponding terminology in the following.

We will first construct the category $\mathbf{RG}_{\text{opt}}$ by filtering the morphisms in $\mathbf{RG}$ via optimal constraints. As such, the objects in the new category remain the same. Suppose that the $N$ samples are enumerated as $\{x_1, x_2, \ldots, x_N\}$, and a grammar $g$ separates the samples into two disjoint classes denoted by $X^g_\text{p}$ and $X^g_\text{n}$. Given two grammars $g_1$ and $g_2$, the morphisms can be seen as a sequence of actions taken to change $X^g_\text{p}$ to $X^{g_2}_\text{p}$. Here the action is to simply flip the label of a sample. Then the optimal constraints imposed on the morphisms only allow ones with the minimal number of actions. We refer to this number as the manipulation distance and denote it as $M_e(g_1, g_2)$. It is easy to check that $M_e(g_1, g_2)$ is equal to the cardinality of symmetric difference $X^g_\text{p} \triangle X^{g_2}_\text{p}$. Note that the morphisms from $g_1$ to $g_2$ are not unique since one can change the order of actions. However, we can identify them as the same since we are only concerned with the manipulation distance instead of the entire transition process.

For the probability distribution category, we can construct $\mathbf{PD}_{\text{opt}}$ by allowing morphisms that only use the Wasserstein edit distance and identifying different transportation plans as the same one, given that one probability distribution can be optimally transferred to another. As for the learning model category $\mathbf{LM}_{\text{opt}}$, we assume that $M_1$ and $M_2$ represent grammars $g_1$ and $g_2$, respectively. Then the morphisms in $\mathbf{LM}_{\text{opt}}$ contain instances of grammar transfer learning with the minimal number of training epochs, which is referred to as the learning distance and denoted as $L_e(M_1, M_2)$. Similarly, different morphisms from $M_1$ to $M_2$ with the same learning distance are considered the same. Note that the optimal categories as defined above are not real categories because the optimal approach from $a$ to $c$ is not the composition of the optimal approach from $a$ to $b$ and $b$ to $c$ in general.

It is easy to notice that functors between these categories naturally hold since the categories are isomorphic to each other. More specifically, the Shapley homology functor, denoted as $\mathbf{SH}$, maps $\mathbf{RG}_{\text{opt}}$ to $\mathbf{PD}_{\text{opt}}$. Also, the learning functor $\mathbf{L}$ maps $\mathbf{RG}_{\text{opt}}$ to $\mathbf{LM}_{\text{opt}}$. Inversely, the transformation from $\mathbf{PD}_{\text{opt}}$ to $\mathbf{RG}_{\text{opt}}$ simply changes a discrete measure space to the set. As such, we refer to it as a measure forgetful functor, denoted as $\mathbf{MF}$. Then
Figure 5.1 Illustration of defined categories and functors.

given an object in $\text{LM}_{\text{opt}}$, one can present all samples $\{x_1, x_2, \cdots, x_N\}$ to a learning model to provide a separation of the set. We denote this map identifying functor from $\text{LM}_{\text{opt}}$ to $\text{RG}_{\text{opt}}$ as $I$. An illustration of the categories and functors is provided in Figure 5.1. In addition, we use the following diagram to summarize the morphisms introduced above:

$$\text{hom}(\text{PD}_{\text{opt}}) \xrightarrow{MF} \text{hom}(\text{RG}_{\text{opt}}) \xrightarrow{L} \text{hom}(\text{LM}_{\text{opt}})$$

Given a morphism $f : g_1 \rightarrow g_2$ in $\text{hom}(\text{RG}_{\text{opt}})$, $M_e$ is defined as $M_e(f) := M_e(g_1, g_2)$. Similar notations can be defined for $W_e$ and $L_e$. Specifically, $M_e$ can be obtained by calculating the cardinality of the symmetric difference of two sets. As for $W_e$, we can obtain its value using linear programming since we are working with discrete measures [140]. Note that $W_e$ degenerates to $M_e$ if one applies the uniform distribution and a constant cost. Also, $L_e$ can be obtained empirically by conducting experiments for grammar transfer learning.

In this way, the problem of examining whether grammar transfer learning is performed in an optimal way is essentially equivalent to checking the property of function $t_1 = t'$ and $t_2 = t \circ t'$. Namely, the non-decreasingness of the functions implies the optimality in grammar transfer learning. Furthermore, we require that the domain of the functions be
the learning distance, and their range to be the manipulation distance and the Wasserstein edit distance. These two metrics provide theoretical optimal distances from two different perspectives: set theory and measure space. In summary, we provide brief procedures for our proposed framework as follows:

1. Generate strings for different regular grammar to construct $\text{RG}_{\text{opt}}$ category;
2. Apply Shapley Homology functor to construct $\text{PD}_{\text{opt}}$ category;
3. Train deep learning models to construct $\text{LM}_{\text{opt}}$ category;
4. Obtain the distances in different categories ($\text{PD}_{\text{opt}}$, $\text{RG}_{\text{opt}}$, $\text{LM}_{\text{opt}}$) by applying optimal transport, symmetric difference and grammar transfer learning, respectively;
5. Compare these values to see their correlation.

Next, we provide experiments to demonstrate the working mechanism of our introduced framework for examining the optimality of grammar transfer learning.

5.4 Experiments

We now introduce our experiment setup and provide the results of investigating the optimality of grammar transfer with discussions. Since this is the first work on studying machine intelligence based on category theory and grammar transfer, our evaluation focuses on examining the effectiveness of our proposed approach.

5.4.1 Experiment Setup

We use a set of seven Tomita grammars [78] to generate the string sets for evaluating transfer performance. We generated the sets of strings by following a prior study [33]. Specifically, for the training sets, we uniformly sampled strings of various lengths $L \in \{0, 1, \ldots, 18\}$ for all seven grammars. For strings from each length, the ratio between the accepted and rejected strings was controlled to be 1:1 where possible, and the number of sampled strings was up to 1,000. We also constructed for each grammar a validation set consisting of up to 500 uniformly sampled strings. Each string has its lengths $L \in \{1, 4, 7, \ldots, 25\}$. On all grammars, we trained a 2-RNN with the same size of the hidden layer ($N_h = 32$) and
initialized it with the same random seed. We monitored the training epochs needed for a model to reach 100% accuracy on the train sets and 99.9+% accuracy on a validation set.

With the generated string sets, we then computed the edit distance between string samples for each grammar and applied Shapely homology to transform the metric space into the measure space. Here we used 0-th Betti number and a resolution of 1 for decomposing the topological features of the space of sample strings. Since it is computationally intractable to apply the Shapley homology framework to a large-scale data set as analyzed in Section 4.4, we compute the spectrum of $K$ subcomplexes instead of all subcomplexes as an approximation, where $K$ is a predefined number (we set $K = 50$ in this work). More specifically, when computing the Shapley value of a specific string, we only randomly selected $K$ subcomplexes that contain this string.

5.4.2 Experiment Results

Here we focus on comparing the Wasserstein edit distance and learning distance while omitting the manipulation distance$^3$. This is because, as we mentioned earlier in Section 5.3.2, the manipulation distance is a special case of the defined Wasserstein edit distances if we neglect the measure information.

In Figure 5.2, we demonstrate with heatmaps the approximated Wasserstein edit distances and the learning distances between all pairs of grammars. The latter is represented by the training epochs needed for transferring an RNN from a grammar to another. Specifically, Figure 5.2a shows the ideal cases since the approximated Wasserstein distances represent the optimal transportation costs between two grammars. Clearly, it is much more difficult to perform the transportation between grammar 1 or 2 and any other grammars since the accepted strings for these two grammars are significantly less than other grammars. Also, we can observe that grammar 5 and 6 are relatively more distinct than other grammars since the costs needed for transporting other grammars to these two (or vice versa) are generally larger. The transportation costs shown in Figure 5.2a are also consistent with the results in Chapter 3 and 4, which shows that the Tomita grammars can be categorized into different classes according to their grammar complexity. In particular, grammar 1 and 2 belong to the same class with the lowest complexity, while grammar 5 and 6 belong to another class with the highest complexity.

$^3$We compute the manipulation distance and provide the results in Figure 5.2c.
Figure 5.2 Heatmaps for the grammar transfer learning performance and the estimated grammar transport learning. In both heatmaps, each entry represents the Wasserstein edit distance/learning distance/manipulation distance from the grammar indicated by its row index to the grammar indicated by its column index.

The empirical results obtained from transferring an RNN from one grammar to another, as shown in Figure 5.2b, however, only partially align with the results shown in Figure 5.2a. We can first observe that the difficulty levels of being learned for grammar 3, 4, 5, and 6 are similar to the results shown in Figure 5.2a. This indicates that a standard gradient descent-based learning algorithm can indeed endow an RNN with a certain level of intelligence that is aligned with ideal intelligence. However, it is also clear that the symmetry in Figure 5.2a
is lost in Figure 5.2b. This is easy to explain since the trained models may only be one of many equivalent sub-optimal solutions. As such, for grammar \( a \) and grammar \( b \), the source \( \text{RNN}_{a\rightarrow b} \) (initially trained on \( a \)) and the target \( \text{RNN}_{b\rightarrow a} \) (finally trained on \( a \)) do not necessarily be the same. Moreover, we can observe that grammar 1 can be easily learned as the target grammar, while grammar 2 is much harder to learn. Note that the 1,100 epochs we show in the three entries in Figure 5.2b represent that the transfer learning has failed in these cases. This 1,100 epoch value is only taken for better scaling the color map used in the heatmap. The reason for the difficulty of learning grammar 2 is that its accepted strings are much too few; hence the training loss will not be significantly affected even when all the accepted strings are misclassified. This effect is also implied by noticing that it is easier to transfer an RNN initially trained on grammar 2 to other grammars. As a result, we can use the inconsistency between Figure 5.2a and Figure 5.2b to highlight that a standard learning algorithm cannot bring real intelligence to a model in a comprehensive way without using external guidance and reference.

5.5 Discussion

As a summary of the Shapley homology and category-based framework, it is essential to point out that these approaches are not only limited to regular grammar. As long as the data space can be topologized by an appropriate metric, we can apply the same framework and procedure to analyze the sample influence and intelligence measurement. For example, \( l_2 \) norm and cosine similarity can be used as a metric for image and text, respectively. Here is an experimental design for the MNIST dataset\(^4\) to demonstrate the universality of the framework. We randomly select 1000 samples from each class and train ten binary classifiers to recognize the digits 0-9, respectively. On the one hand, we can calculate the sample influence and the Wasserstein edit distance between each class in the probability distribution category. On the other hand, we record the number of transfer learning epochs from one to the other as the learning distance. As the main target of this thesis is the connection between formal language theory and deep learning, we will omit the results and analysis of the image dataset here. However, note that different from regular grammar, there is no

\(^4\)The MNIST [173] data set is composed of 70,000 greyscale images (of 28×28, or 784, pixels) of hand-written digits. This data set is split into a training set of 60,000 samples and a test set of 10,000 samples. The classification task is the digit recognition, where the possible digits range from 0 to 9.
canonical correspondence between the image dataset and DNN. The theoretical analysis of DNN representation ability is merely based on the universal approximation theory, which indicates DNN training performed in a manner of approximation. For more discussion, please see Section 6.3.

5.6 Conclusion

We propose a category theory-based framework to investigate the mechanism of grammar transfer learning. We posit that grammar transfer learning can help understand how knowledge in a deep learner can be effectively used and transferred to other machine learning systems. Furthermore, using specific examples of regular grammar can help understand the foundations of transfer learning and reveal actual knowledge learned in the machine and deep learning systems. More specifically, we have constructed three isomorphic categories. Two of which represent optimal transfer approaches from a human perspective. By examining whether there are any correlations between these optimal approaches and grammar transfer learning, one can better understand the degree of intelligence demonstrated by a learning model trained with standard deep learning algorithms. Also, we demonstrated grammar transfer learning on a set of Tomita grammars to illustrate a working mechanism of our proposed framework. Future work could focus on extending this work to more sophisticated problem domains and data sets.
Chapter 6

Adversarial Models for Deterministic Finite Automata and Verification of Recurrent Neural Network

While most prior work contributes to protecting feed-forward networks against adversarial examples, little has been accomplished for RNNs. This is due to more rigorous constraints on the perturbation space for sequential data and the lack of proper metrics for measuring the perturbation. This chapter addresses these challenges by using DFA to represent a rigorous oracle, which examines if the generated adversarial samples violate certain constraints on the perturbation and designing a metric that measures the distance between strings. Specifically, we first investigate a finer-grained understanding of the characteristics of particular deterministic finite automata (DFA). We study and identify the transitions of a DFA that are more important for maintaining the correctness of the underlying regular language associated with this DFA. To estimate transition importance, we develop an approach that is similar to the approach widely used to expose the vulnerability of neural networks with the adversarial example problem. In our approach, we propose an adversarial model that reveals the sensitive transitions embedded in a DFA. In addition, we find for a DFA its critical patterns where a pattern is a substring that can be taken as the signature of this DFA. Our defined patterns can be implemented as synchronizing words, representing the passages from different states to the absorbing state of a DFA. Finally, we have designed a preliminary mechanism to accommodate the rigorous constraints mentioned above for evaluating the
adversarial robustness of several widely used RNNs. To our knowledge, this is some of the
first work to explore adversarial models for DFAs and verification of RNNs.

6.1 Introduction

Verification for neural networks is crucial for validating deep learning techniques in security-
critical applications. However, the black-box nature of neural networks makes inspection,
analysis, and verification of their captured knowledge difficult or near-impossible [174]. More-
over, the complicated architecture of neural networks also makes these models vulnerable
to adversarial attacks [175] – a synthetic sample generated by slightly modifying a source
sample in order to trick a neural network into “believing” this modified sample belongs to
an incorrect class with high confidence.

The adversarial example problem for DNN was initially discussed in a prior study [17],
which is independent of a series of research in the field of adversarial machine learning [16].
The latter field covers a wide range of research on security issues associated with different
stages – training and inference – of the life cycle of a machine learning model. Research on
the adversarial example problem for DNN mainly studies attacks that happen during the
inference stage and defenses that can be deployed in either the training or inference stages
or both.

Most prior work on neural network verification has been on verifying feed-forward neural
networks using [176–178]. Specifically, these approaches can either verify if a neural network
can remain robust to a constrained perturbation applied to an input or approximate the
maximally allowed scale of the perturbation that can be tolerated. To apply these verification
approaches, two critical requirements need to be satisfied. One is that an adversarial sample
should be recognized by a hypothetical oracle that is very similar or even identical to its
source sample. Another requirement is that the adversarial perturbation must be small
enough to avoid being detected by the oracle.

Depending on applications, there are different ways to set up the oracle and various
distance metrics that measure the scale of an adversarial perturbation. For image recognition,
fortunately, it is not challenging to satisfy the two requirements mentioned above. More
specifically, in this scenario, a human is usually assumed to be the oracle, and adversarial
images must avoid a straightforward visual inspection. However, it is neither realistic nor
efficient to assign a human oracle. As such, the oracle, in this case, can be simply replaced

92
by the ground truth labels of source images. As for the distance metrics, various $L_p$ norms ($p = 0, 2, \infty$) have been widely adopted in prior work \cite{175, 176, 179} on adversarial sample problem. The convenience brought by image recognition has made this application the benchmark for much verification work \cite{177, 180–183}.

When dealing with sequential data, e.g., natural language, programming code, DNA sequence, etc., these requirements are challenging to satisfy. This is mainly due to the lack of appropriate oracles and proper distance metrics. For instance, in sentiment analysis, it has been shown that even the change of a single word is sufficient to fool a recurrent neural network (RNN) \cite{184}. However, the adversarial sentence presented in this work \cite{184} contains grammatical errors. This indicates that for sequential data, the adversarial samples need be not only negligible but also satisfy certain grammatical or semantic constraints. Unfortunately, it is very challenging to formulate these constraints and construct an oracle with these constraints. Since RNNs are often used for processing sequential data, the difficulty of verifying sequential data has consequences that limit research work on verifying RNNs.

Here, we propose to use deterministic finite automata (DFA) as the oracle. First, we study individual DFAs for their fine-grained characteristics, including transition importance and critical patterns. Specifically, we examine the importance of transitions by relating this task with the adversarial example problem \cite{185} often seen in deep learning. Because string identification is an important problem in time series, speech, and other scenarios, this motivates research into understanding complicated learning models such as neural networks. One particular approach is to identify feature-level perturbations that significantly affect a learning model. Similar approaches have been used for examining the sample-level importance in building a learning model \cite{84}. These studies use adversarial examples as a data-driven tool for probing the learning model’s vulnerability, hence indirectly gaining an understanding of complicated learning models. In order to directly gain a better understanding of a DFA, we follow a similar approach but study the sensitivity of a DFA through model-level perturbations. Next, we study critical patterns that can be used for identifying a specific DFA. Specifically, we formally define a critical pattern as a substring, which effectively identifies all strings accepted by a certain DFA. We show that for certain classes of DFA, we can identify these strings statistically by checking the existence of critical patterns embedded in their generated strings without exhaustively searching all possible strings or querying the underlying DFA \cite{56, 186}. We then develop an algorithm for finding the critical patterns of a DFA by transforming this task as a DFA synchronizing problem \cite{187}. Last, we provide a
theoretical approach for estimating the length of any existing perfect patterns and validate our analysis with empirical results. Finally, we designed a preliminary mechanism to accommodate the rigorous constraints mentioned above for evaluating the adversarial robustness of several widely used RNNs. We feel that our analysis on DFA models and verification scheme on RNN will help in research on the security of cyber-physical systems that are based on working DFAs, e.g., compilers, VLSI design, elevators, and ATMs. This could be especially for the case when the actual state machine is exposed to adversaries and be attacked. This chapter intends to open a discussion on these issues.

6.2 The Formulation of the Adversarial Example Problem

Given a classification task, we first denote the $d$-dimensional input space as $\mathcal{X} = \mathbb{R}^d$, which contains all possible inputs. This avoids the cases that input data is presented in forms that cannot be processed by a selected classifier. For example, in image recognition, an image needs to be presented to a classifier as a grid or an array of pixels with a size that does not violate the dimensionality requirement. Since we only focus on classification problems in this chapter, we use $\mathcal{Y} = \mathbb{Z}^m$ to represent the label space for $m$-classification problems. As such, a DNN classifier $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ that is built with its associated parameter set $\theta$ maps a data instance $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$. The training set is designated as $D_{\text{train}} = \{X, Y\}$, where $X \in \mathcal{X}^n$, $Y \in \mathcal{Y}^n$, and $n$ denotes the size of the training set. Then we can describe the training algorithm for a $m$-classification problem as $L : (\mathcal{X} \rightarrow \mathcal{Y}) \times (\mathcal{X}^n \times \mathcal{Y}^n) \rightarrow (\mathcal{X} \rightarrow \mathcal{Y})$. Explicitly, the input of $L$ includes $D_{\text{train}}$ and $f_{\theta_0}$, of which the parameters are initialized by $\theta_0$. Then the output of $L$ is $f_{\theta}$, of which the parameters are updated as $\theta$.

For classification problems, the label set $Y$ is obtained by assuming that all training samples have been correctly annotated by some oracle $\lambda$. Here we denote the oracle as a function $\lambda : \mathcal{X} \rightarrow \mathcal{Y}$, which processes any $x \in \mathcal{X}$ and produce the ground-truth classification decision $\lambda(x) \in \mathcal{Y}$. We also denote the space of data classified by $\lambda$ as having the label of $y$ by $\mathcal{X}_y$, i.e. $\mathcal{X}_y = \{x \mid \lambda(x) = y\}$. By expanding upon a formulation that has been commonly used in prior work [17, 188, 189], we provide a more general formulation for the adversarial example problem in the following:
\[
\begin{align*}
\min_{\hat{x}} & \quad D(x, \hat{x}) \\
\text{s.t.} & \quad x, \hat{x} \in X \\
& \quad f_\theta(\hat{x}) \neq f_\theta(x) = \lambda(x) \\
& \quad \lambda(\hat{x}) = \lambda(x).
\end{align*}
\] (6.1)

\(D\) is a distance function selected to measure the difference between an adversarial example \(\hat{x}\) and a legitimate sample \(x\). As will become clear from the discussions presented in the latter part of this chapter, both the objective and the set of constraints in our formulation have rich implications in the characteristics of adversarial examples.

### 6.3 Transition Importance of a DFA

In this section, we aim to extend the original adversarial sample to adversarial model formulation to investigate the transition importance of a DFA. DFAs are one of the simplest automata in the Chomsky hierarchy of phrase structured grammars [6]. Throughout this chapter, all DFAs are complete minimal DFAs. Due to its deterministic nature, it is natural to assume different transitions are equally important for identifying a DFA. However, as will become clear from our analysis, this assumption does not generally hold. Here we illustrate this with the DFA associated with the Tomita-7 grammar shown in Fig. 6.1. Among all transitions, the cyclic transition (with input \(a\)) associated with state-4 is the most important one.\(^1\) This is because by substituting this transition by a transition to state-1 with the same input, we can add significantly more strings to the set of accepted strings.

\[\text{Figure 6.1} \quad \text{Example for the Tomita-7 grammar. Red (Black) states are the accept (reject) states.}\]

\(^1\)We will use \(a, b\) instead of 0, 1 to denote the input alphabet to avoid confusion.
6.3.1 Transition Importance Estimation as an Adversarial Model Problem

To estimate the importance of each transition and identify more important ones, we take an approach that is complementary to the approach used to identify sensitive features of a data sample viewed by a deep neural network (DNN) in the context of the adversarial example problem. As such, the transition importance estimated by our approach essentially reflects the sensitivity of a DFA with respect to a transition.

Equivalent to eq. (6.1), another typical formulation of the adversarial example problem is to maximize a loss function $L$ with respect to a normal sample $x_0$ and a model $f$. Then finding an adversarial sample $\hat{x}$ is conducted by solving the following problem:

$$\hat{x} = \arg \max_{|x-x_0| \leq \epsilon} L(x, f),$$  \hspace{1cm} (6.2)

where $\epsilon$ denotes some predefined constraint on the scale of perturbation. Here we propose to transform the adversarial example problem (eq. (6.2)) into the adversarial model problem, which considers model-level perturbations. Explicitly, given a model $f_0$ and a fixed set of string samples $X$, we try to solve the following problem:

$$\hat{f} = \arg \max_{|f-f_0| \leq \epsilon} \sum_{x \in X} L(x, f).$$  \hspace{1cm} (6.3)

Eq. (6.3) describes the problem of perturbing a target model in a constrained manner to cause maximal loss and provides an alternative view of the adversarial example problem. Specifically, given an ideal mapping $f$ from some functional space $F$ for a certain learning task, and an arbitrarily small approximation error $\epsilon$, the universal approximation theory [190] states that one can always find a candidate $f'$ in some other functional space $F'$ (generally taken as a subset of $F$) satisfying $\|f - f'\| < \epsilon$. Given that DNNs already have very complicated architectures $^2$, we can only measure the difference between $f$ and $f'$ numerically, although these two functions might be quite different. Since these models built through analytical approaches may not necessarily have actions aligning with our intuition and expectation, we cannot easily, if not impossibly, establish a physical understanding of the gap between $f'$ and $f$. Furthermore, in practice, the gap between these two functions may be amplified

---

$^2$Recent research [191] on explaining DNNs have demonstrated the difficulty of analyzing and inspecting these powerful models.
by formulating the approximation problem as an optimization problem and then applying various techniques to solve the latter [192]. These combined effects imply that the root cause of the adversarial example problem lies in both the ambiguity of the theoretical foundation for building a learning model and the imperfection in the practice of applying a learning model. Moreover, it is important to note that our transformation cannot be easily applied to complicated models like DNNs. The function represented by a DNN has too many parameters, including weights, neurons, layers, and all sorts of hyper-parameters. This results in an enormous perturbation space.

On the other hand, for a DFA, the perturbation space is significantly limited to only include its transitions and states. Furthermore, the perturbation of a state can be represented by a set of perturbations applied to the transitions associated with this state. Therefore, in the following, we only consider transition perturbations as they provide a more general description of the adversarial perturbations of a DFA. In addition, we only consider perturbations that make substitution operations on the transitions. This is because for a given DFA, inserting transitions is not allowed since this DFA is already complete and minimal. Also, removing a transition is equivalent to substituting this transition to the transition that connects the current state to an absorbing state, of which the outward transitions all loop back to itself. Our study of the adversarial DFA can be taken as a step in studying the adversarial phenomenon by restricting the underlying models to be physically interpretable and directly investigating the vulnerability of that model.

6.3.2 Transition Importance

The deterministic property of a DFA enables it to be naturally immune to adversarial examples. However, when the adversarial perturbation is applied to a DFA, it is possible to generate an adversarial DFA, which only differs from the original DFA by a limited number of transitions, yet recognizes a regular grammar that is dramatically different from the one associated with the original DFA. To quantitatively evaluate the difference between two sets of strings accepted by different DFAs, here we introduce the following metric:

**Definition 13 (Intersection over Union (IoU)).** *Given two arbitrary DFAs represented by A*
and \( \hat{A} \), and their accepted sets of strings denoted by \( X \) and \( \hat{X} \), respectively, then

\[
\text{IoU}(A, \hat{A}) = \frac{|X \cap \hat{X}|}{|X \cup \hat{X}|}.
\] (6.4)

It is easy to notice that the metric \( \text{IoU} \) is well-defined and lies between 0 and 1. One can apply the L’Hôpital’s rule to calculate it if both the numerator and the denominator approach infinity. By using the above definition of \( \text{IoU} \), we express the adversarial model problem for a DFA as perturbing the transitions of a given DFA to reach a low \( \text{IoU} \). Then we have the following theorem.

**Theorem 6.** Given a DFA with alphabet \( \Sigma = \{a_1, a_2\} \), we use \( A_1 \) and \( A_2 \) to denote its transition matrices associated with the first and second input symbol. Similarly, let \( \hat{A}_1 \) and \( \hat{A}_2 \) denote the transition matrices of perturbed DFA yielding

\[
\text{IoU}(A, \hat{A}) = \left( \sum_{n=1}^{\infty} (1 \otimes p)^T (M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2))^{n}(1 \otimes q) - 1 \right)^{-1}. \] (6.5)

where \( p \in \mathbb{B}^n \) is a one-hot encoding vector to represent the initial state, and \( q \in \mathbb{B}^n \) denotes the set of accept states of a DFA with \( n \) states. We also have \( 1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} , \ M_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} , \ M_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} , \) and \( \otimes \) denotes the Kronecker product.

**Proof.** We use the following Lemma 4 [54] to derive our proof.

**Lemma 4.** Let \( L \) and \( \hat{L} \) be a recognisable language of \( A = \{\Sigma, S, I, F, T\} \) and \( \hat{A} = \{\Sigma, \hat{S}, \hat{I}, \hat{F}, \hat{T}\} \) respectively, then

- the union \( L + \hat{L} \) is recognised by the automaton \( \{\Sigma, S \cup \hat{S}, I \cup \hat{I}, F \cup \hat{F}, T \cup \hat{T}\} \),

- the intersection \( L \cap \hat{L} \) is recognised by the DFA \( \{\Sigma, S \times \hat{S}, I \times \hat{I}, F \times \hat{F}, T'\} \), where \( T' = \{(s_1, \hat{s}_1), \sigma, (s_2, \hat{s}_2)\} \mid (s_1, \sigma, s_2) \in S \) and \( (\hat{s}_1, \sigma, \hat{s}_2) \in \hat{S} , \sigma \in \Sigma \) and \( \times \) denotes the Cartesian product.

For an arbitrary DFA, one can construct an adjacency matrix \( P \) for this DFA by regarding it as a directed graph where each node represents a state, and every edge represents the
existence of a transition between a pair of states. Then from Lemma 4, it is easy to check that for the new automaton that represents the union of two source DFAs, its initial state vector, accepting state vector, and the adjacency matrix are \( 1 \otimes p, 1 \otimes q \) and \( M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2) \), respectively. In this case, the newly created automaton is no longer a DFA since it has two different initial states. As a result, the intersection part is computed twice when calculating the cardinality. Therefore, we will use the Inclusion-Exclusion Principle to correct the calculation. Similarly, for the DFA that recognizes the intersection of two sets of strings accepted by two DFA, its initial state vector, accepting state vector, and the adjacency matrix are \( p \otimes p, q \otimes q \), and \( A_1 \otimes \hat{A}_1 + A_2 \otimes \hat{A}_2 \), respectively.

In order to compute the cardinality of the union set and the intersection set, we need to sum the number of strings of which the length varies from 1 to infinity. Assume that we have two column vectors \( s_I \) and \( s_E \) that represent the set of initial and ending states, respectively. Then the number of \( N \)-length strings that reach \( s_E \) from \( s_I \) is \( s_I^T P^N s_E \). Note that if we calculate the cardinality of the union of two sets of strings as described above, the intersection set of these two sets will be calculated twice. This effect is eliminated by the \(-1\) in eq. (6.5) by Inclusion-Exclusion Principle.

Theorem 6 provides directly an explicit formulation for computing our defined \( IoU \). As such, the original adversarial model problem for the DFA can be transformed into an optimization problem. Furthermore, we require that this manipulation only allows one transition substitution to be applied to one of the transition matrices associated with different inputs. The allowed single transition substitution causes the Frobenius norm of the manipulated transition matrix to be changed by \( \sqrt{2} \). This also avoids changes to the absorbing states of the source DFA (if they exist), so that any existing absorbing states will not be affected. In addition, we require the set of accepted states remains the same. Therefore, we have the following optimization problem \(^3\):

\[
\begin{align*}
\min_{\hat{A}_1, \hat{A}_2 \in T} & \frac{\sum_{n=1}^{\infty} (p \otimes p)^T (A_1 \otimes \hat{A}_1 + A_2 \otimes \hat{A}_2)^n (q \otimes q)}{\sum_{n=1}^{\infty} (1 \otimes p)^T (M_1 \otimes (A_1 + A_2) + M_2 \otimes (\hat{A}_1 + \hat{A}_2))^n (1 \otimes q)} \\
\text{s.t.} & \|\hat{A}_1 - A_1\|_F^2 + \|\hat{A}_2 - A_2\|_F^2 = 2; \\
& y(A_1 + A_2) = y(\hat{A}_1 + \hat{A}_2); \\
& (A_1 + A_2)y^T = (\hat{A}_1 + \hat{A}_2)y^T.
\end{align*}
\]

\(^3\)The constant number 1 is omitted for simplicity.
where \( y = 0 \) when the source DFA does not have an absorbing state and \( y = [0, 0, \cdots, 1] \). Otherwise, \( \mathcal{T} \) denotes the set of transition matrices which contains exactly one 1 in each row, and \( \| \cdot \|_F \) denotes the Frobenius norm.

In practice, it is possible that some additional constraints can be added to the above formulation. Specifically, here we require the perturbed DFA to remain strongly connected, and no new absorbing states will be created. Since it is difficult to formulate these constraints in eq. (6.6), we manually examine their violations in the obtained solutions. Note that these constraints can be easily checked by analyzing the spectrum of the perturbed transition matrix.

### 6.3.3 Evaluation of DFA Transition Importance

In the following, we use the Tomita grammars to demonstrate our estimation of the transition importance of DFAs. Since this is the first work on studying the adversarial scheme of formal computation models, our evaluation mainly focuses on examining the effectiveness of our proposed approach.

In the experiments, we select the Tomita-3/5/7 grammars as examples. These grammars are selected as they are representative of the exponential, proportional, and polynomial classes of regular grammars with the binary alphabet, respectively (These classes are introduced in Section 3.2). Since it is impossible to sum up to infinity, for our evaluation, we fix the maximum length \( N \) of binary strings to 20. Also, instead of solving the original objective which takes a quotient form, we apply the symmetry difference of two sets as an alternative. This choice is reasonable since the objective functions capture the same essence of minimizing the cardinality of the intersection and maximizing the cardinality of the union of two sets. Furthermore, as the original problem is formulated as a high-order integer programming problem, which is difficult to solve with existing solvers, we relax the constraints such that \( \hat{A}_1 \) and \( \hat{A}_2 \) are constrained as row-wise stochastic matrices as their entries. As such, we determine the final perturbation by selecting the one with the maximal value, which represents the maximal transition probability. We notice that our approximation may not yield the real optimal solution; however, as shown by the results in Table 6.1, it provides satisfying results in analyzing the transition importance.

The effectiveness of our optimization approach when comparing it with a randomization approach is shown in Table 6.1. Specifically, for the randomization approach, we randomly
Table 6.1 Optimization results for the Tomita-3/5/7 grammars.

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Value from optimization</th>
<th>Value from randomization</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.48e-3</td>
<td>0.342</td>
</tr>
<tr>
<td>5</td>
<td>0.152</td>
<td>0.289</td>
</tr>
<tr>
<td>7</td>
<td>0.025</td>
<td>0.225</td>
</tr>
</tbody>
</table>

(a) Grammar 3
(b) Grammar 5
(c) Grammar 7

Figure 6.2 Illustration of identified important transitions for example DFAs. The marked (with a yellow cross) and dashed lines demonstrate the most sensitive transitions of the original DFAs and the perturbed transitions, respectively.

select five legitimate perturbations (manually checked according to the constraints described above) and calculate and average the resulting \( \text{IoU}_{\text{rand}} \). We then compare \( \text{IoU}_{\text{rand}} \) with the \( \text{IoU}_{\text{opt}} \) obtained by our optimization. It is clear that the results provided by the optimization approach are much more desirable. We also provide a visualization of the perturbations generated by our approach for each investigated grammar in Fig. 6.2.

6.4 Critical Patterns of DFA

6.4.1 Different Types of Critical Patterns

Here, we provide a relatively coarse-grained view, in contrast to what we described regarding transition importance, to investigate the characteristics of a DFA. Specifically, we identify critical patterns of a DFA, defined as:

Definition 14 (Absolute and relative patterns of a DFA). Given the alphabet \( \Sigma \) of a DFA and a data space \( \mathcal{X} \subseteq \Sigma^* \), \( \mathcal{X} \) is the union of two disjoint sets, i.e., \( \mathcal{X} = P \cup N \), and we

\( \text{for a DFA, } P (N) \) represents the space of strings accepted (rejected) by this DFA.
Figure 6.3 An illustration of the difference between absolute and relative patterns.

define the following patterns:

\[
\hat{m} = \arg \max_{|m|=k} \left| Pr_{m \sim f y}(y \in P) - Pr_{m \sim f y}(y \in N) \right|. \quad (6.7)
\]

\[
\hat{m} = \arg \max_{|m|=k} \left| Pr_{y \in P}(m \sim f y) - Pr_{y \in N}(m \sim f y) \right|, \quad (6.8)
\]

where \( y \) is a string in \( X \) and \( m \sim f y \) indicates that \( m \) is a factor (consecutive substring) of \( y \).

Here we focus on the general case where all the strings follow the uniform distribution without using any particular prior knowledge. We illustrate the difference between the absolute and relative patterns with the example in Fig. 6.3 by splitting the entire data space \( X \) into four parts denoted as \( \{a, b, c, d\} \). According to eq. (6.7), an absolute pattern describes the substring \( m \) (has the length of \( k \)) such that, among all strings that contain \( m \), it causes the largest discrepancy between the probabilities of a string that belongs to different disjoint sets. Thus, the absolute pattern differentiates strings in \( \{a, b\} \), and the objective in eq. (6.7) are equal to \( \frac{|a-b|}{a+b} \). In contrast, a relative pattern is identified by considering the statistics of the entire data space, with the objective in eq. (6.8) equal to \( \left| \frac{a}{a+c} - \frac{b}{b+d} \right| \). Note that these two patterns are equivalent to each other under certain
circumstances. For example, consider a DFA that rejects any binary string containing “$bbb$” as a substring. In this case, both the absolute and relative patterns identify the factor “$bbb$”. Here, we are concerned with the absolute pattern since it provides better insight into the connection between identified patterns and the underlying DFA. In contrast, a relative pattern mainly provides a conceptual understanding from a statistical perspective. Furthermore, we introduce the following definition:

**Definition 15 (Perfect absolute pattern of a DFA).**

Let $A_p = \{ m \mid \max_m |Pr_{m \sim y}(y \in P) - Pr_{m \sim y}(y \in N)| = 1\}$, then the perfect absolute pattern is defined as:

$$\hat{m} = \arg\min_{m \in A_p} |m|.$$  \hfill (6.9)

A perfect absolute pattern describes a substring, which has minimal length among all absolute patterns and perfectly differentiates the strings from different disjoint sets. However, not all DFAs have perfect absolute patterns. Some DFAs, which have a cyclic property, contain recurrent or persistent states that contain both accepting and non-accepting states. This indicates that these DFAs can never determine the label of a string until they finish processing the entire string. These DFAs with a binary alphabet, as previously determined in Section 3.2, belong to one of three classes, which are then categorized according to the complexity of different DFAs. Upon inspection, it is just a random guess for identifying a string accepted by a DFA which has no absorbing state and by only checking its contained factors. Also, determining the pattern of a DFA, which contains two absorbing states, can be taken as performing a random guess twice. As such, we only focus on the following analysis on DFAs belonging to the polynomial and the exponential classes. Importantly, we find that identifying a perfect absolute pattern of a DFA is essentially analogous to designing a synchronizing word \[187\] for the absorbing state of a DFA. Therefore, instead of solving the optimization problem in eq. (6.7), we propose a DFA synchronizing word approach and design a metric to evaluate the confidence for determining whether a certain string belongs to a particular class. We show that our metric is highly correlated with the probability in eq. (6.7).

---

\(^5\)The example DFA is associated with the Tomita-4 grammar.
6.4.2 DFA Synchronizing Algorithm

Recall that the synchronizing word (or the reset sequence) is a substring that sends any state of this DFA to the same state. An absorbing state naturally fits this synchronizing scheme. As such, we can set the absorbing state as the state to be synchronized. And since all states will result in an absorbing state when applying the same substring to these states, the label of a string containing the substring can definitely be determined.

However, given a string of fixed length \( k \), there is no guarantee that we can always reach an absorbing state. Thus, we design the following algorithm and metric to evaluate the efficiency of identifying an absolute pattern. More specifically, given a DFA with \( n \) states and a predefined length \( k \), we then have its \( k \)-order transition matrix \( A_k \) by multiplying the transition matrix \( A \) by itself \( k \) times. We focus on the column associated with the absorbing state. This column represents the prefixes coming from all states to this absorbing state. We now choose the most frequent substring \( m \) appearing in this column. We denote that number of occurrences as \( \hat{n} \) and determine \( m \) since an absolute pattern has the confidence of \( \hat{n}/n \). For the perfect absolute pattern, the confidence is 1 since the substring sends all other states to the absorbing state, and it will appear in each entry of the column associated with the absorbing state of the \( k \)-order transition matrix. In experiments presented in the latter part of this section, we demonstrate the results of applying this algorithm.

Furthermore, given a DFA with one absorbing state, similar to the Černý's conjecture [193], we can estimate the length of a perfect absolute pattern associated with a DFA by providing a loose upper bound. That is, given a DFA with an absorbing state, we have the following theorem for estimating the minimal length of a synchronizing substring, which leads all states to the absorbing state.

**Theorem 7.** The length of a perfect absolute pattern of a DFA with \( n \) states is at most \( n(n - 1)/2 \).

**Proof.** In order to obtain an upper bound of the length of a perfect absolute pattern, we need to consider the worst case of synchronizing the absorbing state of a DFA. Specifically, in the worst case, as the states of a DFA are connected, the distance between each state and the absorbing state is 1, 2, \( \cdots \), \( n - 1 \) respectively. This also requires that the process starts by synchronizing the states that are nearest to the absorbing state. Therefore, at the first step of synchronization, the substring connecting the current state and the absorbing state has a length of 1. Note that the worse case also indicates that, at step \( t \), synchronizing the nearest
state is the optimal choice. Furthermore, after synchronizing the \( t \)-step nearest states, the distances between the rest of \( n - t \) states and the absorbing state range exactly from \( t + 1 \) to \( n - 1 \) during this iterative process.\(^6\) In this case, at the \( t \)-step of synchronization, the substring will have length \( t \). As such, to synchronize all states in the worst case, we have the length of a synchronizing substring at most \( 1 + 2 + \cdots + (n - 1) \), which is equal to \( n(n - 1)/2 \).

It is straightforward to check that the upper bound in Theorem 7 holds for any size of the alphabet. As such, we conjecture that there exists a tighter upper bound, which depends on the number of states and the DFA alphabet size. Next, we provide some examples in order to further investigate the pattern length.

We demonstrate in Fig. 6.4a and Fig. 6.4b that when the number of states of a DFA is set to 3 or 4, we can construct a DFA for which the perfect absolute pattern meets the upper bound exactly. Specifically, for the DFAs shown in Fig. 6.4a and Fig. 6.4b, their associated patterns are \( bab \) and \( babaab \). However, it is impossible to construct a 5-state DFA, for which the perfect absolute pattern has a length that reaches the upper bound in Theorem 8. More specifically, we have the following result:

**Theorem 8.** The length of an absolute pattern of a 5-state DFA is at most 9.

**Proof.** This theorem can be proved by using combinatorics and enumeration techniques. Suppose that we can construct a 5-state DFA with a length of absolute pattern 10. First, let state 5 be the absorbing state. By simple observation, we must have the following DFA transitions to achieve the upper bound: \( 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \), and there is no transition from state \( i \) to \( i + k \) for \( k \geq 2 \). Without loss of generality, we assume the transition from

\[\begin{array}{c}
\text{1} \quad \text{a} \\
\text{a} \quad \text{b} \quad \text{a} \\
\text{a} \quad \text{b} \\
\text{a} \quad \text{b} \\
\text{b} \quad \text{a} \\
\end{array}\]

Figure 6.4  DFA examples that illustrate Theorem 7 and 8.
state 4 to 5 is $4 \rightarrow 5$, then the first letter of the synchronizing word must be $b$, and in this way, the transition map $b : \{1, 2, 3\} \rightarrow \{1, 2, 3\}$ must be injective and surjective. Hence we have $3 \rightarrow 4$. And the second step is to synchronize state 3, which indicates that we have pattern $bab$ as the first three letters of the synchronizing word. Furthermore, the remaining states to be synchronized must be 1 and 2 after applying the transition $bab$. In this way, we discuss the following three possibilities:

If $3 \rightarrow 3$, so $2 \rightarrow 3$. As $bab$ is not allowed to send state 2 to 3, so we must have $2 \rightarrow 1$, then $1 \rightarrow 2$. In this case, $bab$ will send state 1 to 3. Contradiction.

If $3 \rightarrow 2$, as there is no transition from state 1 to 3, we must have $1 \rightarrow 1, 2 \rightarrow 3$ and $1 \rightarrow 2$. Then $bab$ will send state 1 to 3. Contradiction.

If $3 \rightarrow 1$, then we have $1 \rightarrow 2, 2 \rightarrow 3$. As $bab$ is forbidden to send state 3 to itself, so $1 \rightarrow 1$. Hence we need $bab$ sends state 1 to itself, which gives $2 \rightarrow 3$. The only transition left is $4 \rightarrow ?$, however, it does not matter at all. This case is shown in Figure 6.4c, which only has a pattern with a length of 9.

\[\square\]

6.4.3 Evaluation of DFA Pattern Identification

In the following experiments, we use the Tomita-4 and Tomita-7 grammars (indexed as DFA-1 and DFA-2), which are representative grammars for the exponential and the polynomial classes, respectively, and also use randomly generated other DFAs as shown in Table 6.2. For all DFAs, we set their starting state as state 1 and their absorbing states as the states with the largest indexes. By applying the algorithm we previously introduced, we obtain and demonstrate in Table 6.3 identified patterns for all evaluated DFAs.

<table>
<thead>
<tr>
<th>Table 6.2</th>
<th>Transition matrices for example DFAs. $A_\sigma(s_1, s_2)$ represents a transition from state $s_1$ to $s_2$ via $\sigma$. We only provide the transitions matrices for DFA-3/4/5 and omit the matrices for DFA-1/2 (Tomita-4/7).</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFA 3</td>
<td>$A_0(1, 1)$ $A_0(2, 2)$ $A_0(3, 3)$ $A_0(4, 4)$ $A_0(5, 5)$ $A_0(6, 6)$</td>
</tr>
<tr>
<td>DFA 4</td>
<td>$A_0(1, 1)$ $A_0(2, 2)$ $A_0(3, 3)$ $A_0(4, 4)$ $A_0(5, 5)$ $A_0(6, 6)$</td>
</tr>
<tr>
<td>DFA 5</td>
<td>$A_0(1, 1)$ $A_0(2, 2)$ $A_0(3, 3)$ $A_0(4, 4)$ $A_0(5, 5)$ $A_0(6, 6)$</td>
</tr>
</tbody>
</table>

106
Table 6.3  Patterns and their corresponding confidence for example DFAs. When several patterns have the same length, we randomly show only one of them.

<table>
<thead>
<tr>
<th>Length</th>
<th>DFA 1</th>
<th>DFA 2</th>
<th>DFA 3</th>
<th>DFA 4</th>
<th>DFA 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>bb</td>
<td>2/3</td>
<td>ab</td>
<td>3/5</td>
<td>0.674</td>
</tr>
<tr>
<td>3</td>
<td>bbb</td>
<td>1</td>
<td>bab</td>
<td>4/5</td>
<td>0.912</td>
</tr>
<tr>
<td>4</td>
<td>abab</td>
<td>1</td>
<td>ab</td>
<td>1/2</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>baabb</td>
<td>2/3</td>
<td>aabb</td>
<td>5/7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>bhaabb</td>
<td>1</td>
<td>aaabb</td>
<td>5/7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>aaaaab</td>
<td>5/7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>bhaaaaab</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

We observe in Table 6.3 that our algorithm successfully identified the perfect absolute pattern for the Tomita-4 (DFA-1) grammar. We also find that the length of an identified perfect absolute pattern does not necessarily increase as the number of states increases. Moreover, we observe that the confidence for determining an absolute pattern for all DFAs is non-decreasing as the length of the identified pattern increases. To further understand the relationship between the confidence and the probability introduced in the definition of the absolute pattern, we design the following experiment and use DFA-2 as our demonstrative example. Specifically, we generate 1000 strings for each identified pattern with their lengths less than 15. We then calculate the frequency of the generated strings that appear in both the accepted and rejected sets, respectively. In particular, we use that frequency to approximate the probability by using the law of large numbers. We show in Table 6.3 that the probability difference and confidence have a positive correlation. Although we do not establish a theoretical relationship between the above-mentioned two statistics, we empirically show that it is reasonable to replace the probability with the confidence. As such, we believe these results validate the effectiveness of our algorithm.

6.5 Verification of Recurrent Neural Network

We have investigated adversarial models for DFAs in previous sections. In this section, we aim at exploiting extracted rules, which are represented by deterministic finite automata (DFA), for evaluating the adversarial robustness of RNNs. Rule extraction is commonly seen in prior work that attempts to establish both theoretical and empirical connections between RNNs and finite state machines [33,34,59,144]. Such connections had been proposed early on
by Minsky [194]. Specifically, if one treats the information processing of a recurrent network as a mechanism for representing knowledge in the symbolic form, where a set of rules that govern transitions between symbolic representations are learned, then the behaviors of an RNN can be viewed as an automated reasoning process with production rules, which should be easier to understand.

We first denote the domain of all strings as $X = \Sigma^*$, where $\Sigma$ is the alphabet for these strings. Then we denote an oracle by $\lambda$, which can process any $x \in X$ and produce a classification decision $\lambda(x) \in \mathcal{Y}$. The set of strings classified by $\lambda$ having the same label of $y$ is denoted by $X_y$, i.e. $X_y = \{x \mid \lambda(x) = y\}$. We assume there is a distance metric denoted by $D$ to measure the distance between strings. Let $\mathcal{P}(x)$ denote the set of all possible strings generated by perturbing $x$ with respect to a certain distance constraint, i.e., $\mathcal{P}(x) = \{x' \mid D(x, x') \leq D(x, \mathcal{X}_\mu)\}$, where $\mu \in \mathcal{Y} \setminus \lambda(x)$, represents any label that is different from $\lambda(x)$. The above constraint indicates that the allowed perturbation to $x$ must not lead to a different classification result made by $\lambda$. Similarly, a RNN $f$ can process any $x \in X$ and produce a vector of classification scores, i.e. $f(x) \in \mathbb{R}^{||\mathcal{Y}||}$ and $\sum_{i=1}^{||\mathcal{Y}||} f_i(x) = 1$. Then we say $f$ is robust or locally invariant [195] with respect to $x$ if and only if finding a $x'$ that satisfying (6.10) is infeasible:

$$
(x' \in X \cap \mathcal{P}(x) \cap \mathcal{X}_{\lambda(x)}) \land (f_{\lambda(x)}(x') < \max_{\mu \in \mathcal{Y} \setminus \lambda(x)} f_{\mu}(x'))
$$

(6.10)

To describe the relation between $f$ and $\lambda$ from a global perspective, we adapt the local invariance property described above to determine the equivalence [195] between $f$ and $\lambda$. More formally, we say there exists an equivalence relation between $f$ and $\lambda$ if it is infeasible to find a $x$ that satisfies the following:

$$(x \in X) \land (\arg\max_{\mu \in \mathcal{Y}} f_{\mu}(x) \neq \lambda(x)).$$

(6.11)

It is clear that $\lambda$ and $D$ play two crucial roles in the above framework for evaluating the robustness of an RNN. As such, it is important to have some $\lambda$ with high “quality” to represent the oracle. In much existing research [27, 33, 59, 196], using DFA as rules extracted from RNNs has been prevalent. In these studies, an RNN is viewed as representing the state transition diagram of a state process – \{input, state\} $\Rightarrow$ \{next state\} – of a DFA. Correspondingly, a DFA extracted from an RNN can globally describe the behavior of this
Algorithm 3 Evaluation of the Adversarial Robustness of a RNN

Input: RNN $f$; Extracted DFA $r$; String length $N$; Number of samples $M$; Allowed perturbed distance $d$;
Output: Adversarial accuracy $\gamma$;

1: Randomly generate $T$ samples $X$ with length $N$, and $X = \{x_i \mid r(x_i) = f(x_i) = p\}$, where $p$ denotes the positive label;
2: $count \leftarrow 0$;
3: for $i = 1$ to $M$ do
4:   Generate samples $O_{x_i}$ from $x_i$ satisfying $O_{x_i} = \{x_j \mid d_e(x_i, x_j) \leq d\}$;
5:   for $j = 1$ to $|O_{x_i}|$ do
6:     if $r(x_j) = n$ then
7:       Continue;
8:     else if $f(x_{ij}) = n$ (where $n$ denotes the negative label) then
9:       $count \leftarrow count + 1$;
10:      Break;
11:    end if
12:   end for
13: end for
14: $\gamma \leftarrow 1 - count/N$;
15: return $\gamma$;

RNN. On the other hand, we apply edit distance introduced in Section 3.5 as the metric $D$. At last, in Section 6.6, we develop an algorithm that exploits extracted DFA and edit distance for evaluating the adversarial robustness of different recurrent networks trained for learning different DFAs.

6.6 Evaluation of the Adversarial Robustness of Recurrent Networks

In this section, we propose to integrate extracted DFA as the oracle and the edit distance as the distance metric into the evaluation framework introduced in eq. (6.10), and apply this mechanism to several widely used RNNs on the Tomita grammars. For the stability of DFA extraction from RNNs, please refer to [59]. We demonstrate that using DFA can evaluate the adversarial accuracy of different RNNs. The experiments show the difference between the robustness of RNNs and the difference in the difficulty of robust learning of grammars with different complexity.
Following the framework described in eq. (6.10), we present in the following experiment the results of verifying RNNs with DFAs. It is important to note that when selecting a ground-truth DFA as the oracle, we can examine the robustness or adversarial accuracy of a certain RNN with respect to small-scale perturbations. If we select an extracted DFA as the oracle, then our evaluation framework can be adopted for examining the fidelity of the extracted DFA. The latter case will be studied in our future work.

Given a well-trained RNN and a ground truth DFA associated with the grammar used for training this RNN, our task mainly focuses on the local invariance property [195] of the RNN. More specifically, we examine the case when a small-scale perturbation is applied to a positive string $x$, whether an RNN will produce a negative label while the DFA still accepts $x$. We also report the results for evaluating the local invariance property of an RNN with negative strings. Here we have only used grammar 3, 4, and 7 for the evaluation task. This is because, for other grammars, it is easy to check that given a positive string $x$, almost all strings with the edit distance to $x$ equals 1 belong to the negative class. This means that for grammar 1, 2, 5, and 6, all their positive samples lie on the decision boundary; hence the perturbation space is rather limited. While for grammar 3, 4, and 7, it is easier to find adversarial samples with a small perturbed edit distance. As such, in the following experiments, we set the maximal allowed perturbed edit distance as 1.

Since all RNNs were sufficiently trained on short strings that make up the training and test sets, we evaluated these models’ adversarial accuracy on long strings. We randomly sampled strings with length 200 to construct the evaluation data sets. All sampled strings were examined to be correctly classified by both a target RNNs and the ground truth DFA for grammar 3, 4, and 7. Since the number of strings increases exponentially as their length increases, we randomly sampled 100 positive strings and 100 negative strings for 30 trials in the evaluation. This allowed us to better approximate the ideal results by exploring the data space. Based on eq. (6.10), we have designed an algorithm for a single trial evaluation, as shown in Algorithm 3.

The $\gamma$ obtained from 30 trials of evaluating positive (negative) strings were averaged and denoted as $\bar{\gamma}_+$ ($\bar{\gamma}_-$). The results presented in Table 6.4 indicated the different levels of robustness obtained by different RNNs. In particular, 2-RNN and MI-RNN were most robust, with no adversarial samples identified, while other RNNs suffered from adversarial samples to a different extent. These results are consistent with the results reported previously in Section 3.5. Of note, the Elman-RNN obtained the lowest adversarial accuracy when
Table 6.4  Verification results for positive and negative strings with the length of 200.

<table>
<thead>
<tr>
<th>G</th>
<th>( y )</th>
<th>( \gamma )</th>
<th>2nd</th>
<th>Elman</th>
<th>MI-RNN</th>
<th>GRU</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>( \gamma^+ )</td>
<td>1.00</td>
<td>( 3.96 \times 10^{-2} )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>( \gamma^- )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>( \gamma^+ )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>( \gamma^- )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>( \gamma^+ )</td>
<td>1.00</td>
<td>( 0.99 )</td>
<td>1.00</td>
<td>( 0.99 )</td>
<td>( 0.98 )</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>( \gamma^- )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Figure 6.5  Adversarial accuracy of an Elman-RNN for strings with varying length (from 100 to 200) on grammar 3.

evaluated for the positive strings from grammar 3. To understand the reason for this worst result, we show in Figure 6.5 the changes of the Elman-RNN in terms of its adversarial accuracy when the length of strings sampled for evaluation was changed. The result indicated that an Elman-RNN could not generalize to long strings and might cause it to suffer from adversarial attacks more likely.

The difference between RNNs’ robustness against adversarial samples, as shown in Table 6.4, could result from a difference between the underlying grammars. Specifically, grammar 4 has demonstrated better robust learning than grammar 3, even though these two grammars have similar levels of complexity. As for grammar 7, although it has the lowest complexity compared to grammar 3 and 4, our algorithm has identified effective adversarial
samples for most RNNs. This indicates that grammar 7 is prone to overfitting the RNNs since the data sets for this grammar are imbalanced.

6.7 Conclusion

In this chapter, we studied adversarial samples in a broader context. In summary, we first defined transition importance and critical patterns for DFAs, which we believe gives insight into understanding and identifying specific DFAs. Specifically, we transformed the widely accepted adversarial sample scheme to an adversarial model scheme, revealing the sensitivity of a model concerning its components. For the case of a DFA, we focus on the components represented by its transitions. In addition, we have designed an effective synchronizing algorithm to find critical patterns of a DFA and studied the upper bound of the length of a perfect absolute pattern. Then we proposed to verify recurrent networks with DFA extraction. We extend the verification framework proposed in prior work for feed-forward neural networks to accommodate the rigorous requirements for the verification of recurrent networks. At last, we have developed an algorithm under our evaluation framework and conducted a case study of evaluation for several different RNNs on the Tomita grammars. The results demonstrate that while all RNNs can sufficiently learn short strings generated by the different Tomita grammars, only certain RNN models can generalize to long strings without suffering from adversarial samples. As such, in our future work, we consider a practical way of integrating extracted DFAs as the oracle is to build an ensemble of extracted DFAs by following a previous study [197], which ensembles a set of simple classification models then combine and weight their decisions to provide an improved classification result.
Chapter 7

Conclusion and Future Work

In this chapter, we will briefly summarize the research achievements and propose future work.

7.1 Conclusion

As the research and application of deep neural networks (DNNs) proceed dramatically, the concern with the robustness of these powerful models increases. This concern is manifested in the intensive research on the adversarial sample problem and explainable artificial intelligence. On the other hand, the resurgence of grammar learning with machine learning techniques requires us to revisit the formal language theory from a new perspective. Therefore, this dissertation aims to build a connection between formal language theory and deep learning.

In this dissertation, we first presented theoretical analysis and empirical validation to investigate the internal structure of regular grammars. We studied the fine-grained structure through the lens of machine learning, i.e., we aim to build a close connection between subclasses of regular grammars and machine learning tasks. Specifically, motivated by the concentric ring representation of regular grammar, we conducted a complexity analysis and proposed an entropy metric to reflect the difficulty in training RNNs to learn the grammar. The regular grammar can be categorized into three disjoint subclasses: polynomial class, exponential class, and proportional class based on the entropy values. Furthermore, we provided classification theorems for different representations of regular grammar. By evaluating and comparing various recurrent networks on different regular grammars, we demonstrate
that regular grammar with higher entropy has more difficulty being learned by neural networks. Also, this work provides insights into understanding and analyzing regular grammar and recurrent networks. The importance of this work is that it is the first research to our knowledge that focuses on the internal structure of regular grammar in terms of machine learning tasks. Furthermore, our work can be viewed as an attempt towards revealing the mystery of the success of recurrent models by discussing the relationship between grammar learning and complexity. In addition, it provides a preliminary discussion of the connection between our proposed entropy and symbolic dynamics.

In the second thread of this dissertation, we focus on a model-independent framework of sample influence, which is currently an important topic in adversarial machine learning and explainable artificial intelligence (XAI). Specifically, we propose the Shapley Homology framework to study the sample influence on topological features of data space and its associated entropy. This provides an understanding of the intrinsic properties of both individual samples and the entire data set. Under our proposed framework, we design two algorithms for decomposing the 0th and 1st Betti numbers, respectively using cooperative game theory, and provide analytical results for several special representative topological spaces. By interpreting the influence as a probability measure, we further defined an entropy that reflects the complexity of the data manifold. Next, we empirically verify our results with two carefully designed experiments. We show that data points identified by our algorithm as having a larger influence on the topological features of their underlying space also have more impact on the accuracy of neural networks in determining the connectivity of their underlying graphs. Also, we demonstrate that regular grammar with higher entropy has more difficulty of being learned by neural networks. We believe these results will provide guidance for others who are studying sample influence in real-world data and on how to perform theoretical analysis from a geometric view. Furthermore, by considering Shapely homology as a functor from regular grammar category to probability distribution category, we proposed a category-based framework to reveal the mechanisms of grammar transfer. This work can be viewed as a fundamental application of algebraic topology that expands the original scope of topological data analysis. In addition, it provides a preliminary discussion of the connection between Shapley Homology and the Vapnik-Chervonenkis (VC) dimension. At last, we believe the renaissance of general intelligence should not be compromised with the upgrade of hardware or the development of computational power. We initiate our attempt at improving the understanding of artificial intelligence by embedding our philosophical thinking into a
formal test method.

Our third thread of study focused on the adversarial example problem applied to the recurrent neural network (RNN) and tried to approach this problem from a more general perspective. We first highlighted the difference between creating adversarial examples on recurrent networks and feed-forward networks. Then we applied the idea of the adversarial sample problem to study the fine-grained characteristics of DFA. In other words, we conducted a thorough analysis of DFA by extending the original formulation to a more general scheme for adversarial models. Specifically, we mainly investigate the transition importance of a DFA through a model-level perturbation and critical patterns that can be used for identifying a specific DFA. In addition, we develop an algorithm for finding the critical patterns of a DFA by transforming this task as a DFA synchronizing problem and provide a theoretical approach for estimating the length of any existing perfect patterns. Furthermore, we proposed a generic framework for evaluating the adversarial robustness of RNN by introducing an oracle and a proper distance metric. At last, we developed an algorithm under our designed evaluation scheme and conducted a case study of verification for several different RNNs on a set of regular grammars. Since it is very difficult, if not impossible, to design comprehensive rules and distance metrics for real-world sequential data, we propose our last thread of work as a stepping stone for estimating the adversarial robustness of RNNs that can be built for more sophisticated applications and can have extracted rules.

7.2 Future Work

Fine-grained Analysis of Other Types of Grammars  In Chapter 3 and 6, we applied concentric ring representation and adversarial model scheme to carefully investigate the fine-grained characteristics of regular grammar and DFA respectively. Note that we could extend the same idea to other types of formal languages and computational models. Specifically, the concentric ring representation is independent of grammar type and can be similarly applied to context-free grammars giving similar results to their entropy. What we need to pay attention to is the classification results due to the fact that context-free grammar does not maintain rationality property any longer. For instance, when we consider the parentheses grammar, the entropy can be obtained by the central binomial coefficient, which fails to fall in any of the classes proposed in this work. Similarly, we could also adopt an adversarial model to study the sensitivity of push-down automata. Unlike DFA, the perturbation space
of push-down automata will be infinite, which is the central issue to be addressed. Future work also extends these proposed approaches and frameworks to more sophisticated problem domains, complex models, and data sets.

Sample Complexity of Stateful and Stateless Models  Generalization bounds for neural networks present an active field of research dedicated to understanding what drives their empirical success from a theoretical perspective. In our designed Shapley homology framework introduced in Chapter 4, we have a brief discussion on the statistical learning theory of neural networks. Future work will focus on studying the generalization ability of different models representing regular grammar. Specifically, we could consider two crucial representations of regular grammars: sets of strings and graphs. Given regular grammar, one can construct a topological space by pair-wise edit distances and transform the space into a graph by simplification. Hence, the two different representations can be learned by two different machine learning models: stateful and stateless models, respectively. By comparing the sample complexity for two representations, we could obtain which one is superior in terms of the number of required samples.

Refinement for Recurrent Network Through Rule Extraction  In this line of future work, we would employ a DFA-based verification for model refinement. Specifically, since a DFA is usually much easier for formal analysis, we could efficiently identify certain implicit weaknesses of a recurrent network by using a DFA extracted from that RNN to generate specific adversarial examples. The generated adversarial examples could then be used for refining the source RNN. Also, we could design different regularization methods by exploiting either extracted DFA or the quadratic interaction taken by second-order RNN for training recurrent networks to combine desirable performance with reliable rule extraction.
Bibliography


odds in Language Processing and Computational Natural Language Learning, NeMLaP/CoNLL 1998, Macquarie University, Sydney, NSW, Australia, January 11-17, 1998, 1998, pp. 73–78.


Vita

Kaixuan Zhang

Kaixuan Zhang enrolled in the Ph.D. program in Information Sciences and Technology at Pennsylvania State University in August 2016. Prior to that, he received his B.S. and M.S. degree in Electrical Engineering from Harbin Institute of Technology from China in June 2016 and 2014, respectively. His research interests include automata theory and adversarial machine learning. His work has led to 9 peer-reviewed papers published in conferences and journals. Here several main papers used in this dissertation are listed below.

Selected Publications:


