creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학박사 학위논문

Robust Feature Learning on Sequential Data

순차 데이터에 대한 견고한 특징 학습

2018년 8월

서울대학교 대학원

전기·정보공학부

권 선 영

Robust Feature Learning on Sequential Data

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함

2018년 8월

서울대학교 대학원

전기·정보공학부

권 선 영

권선영의 박사 학위논문을 인준함

2018년 8월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

to my family

Abstract

A sequence can be defined as a series of related things: a series of words in natural language, symbols in biological sequences, or a time series representing a stock market or biological signal. Drugs can also be expressed as a series of symbols representing chemical properties. Values in a sequence are not assumed to be independent and identically distributed (i.i.d.), but rather have local or long-distance dependencies and hidden patterns of variable length.

This dissertation proposes three methodologies to understand hidden dependencies and patterns in sequential data. First, motivated by the demand for the accurate, fast, and scalable classification of nucleotide sequences against a group of sequences, we propose a new generative classification method based on the variable-order Markov model and universal probability. Second, to learn chemical features automatically without manual curation in chemical-chemical interaction (CCI), we propose an end-to-end convolutional neural network (CNN) based on Siamese architecture. Finally, to achieve robust results in quantitative structure activity relationship (QSAR) prediction, we propose comprehensive ensemble learning incorporating an individual classifier with CNNs and recurrent neural networks (RNNs).

In summary, this dissertation describes methodologies for solving diverse sequential problems. Information theoretic approaches, and neural network of CNN and RNN approaches were used exquisitely, and we confirmed their robust performance.

keywords: machine learning, deep learning, neural network, information theory, sequence classification, drug analysis

student number: 2014-31068

Contents

Abstract **i**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**

2 Background **8**

2.1 Sequential Data 8

2.2 Representation of Sequential Data 11

2.2.1 Nucleic Acid Sequence 11

2.2.2 Chemical Compound 15

2.3 Preprocessing of Sequential Data 19

2.4 Methods for Processed Sequential Data 21

2.4.1 Information Theoretic Approaches 21

2.4.2 Convectional Machine Learning Approaches 24

2.4.3 Deep Neural Network Based Approaches 24

3 Variable-order Markov Model Based Sequence Classifier **28**

3.1 Methods 31

3.1.1 Model building 33

3.1.2 Classification 35

3.1.3 Outlier detection in a family of sequences 36

3.1.4 Synthetic sequence generation from learned CTM 37
3.2 Results 37
3.2.1 Dataset preparation and experimental setup 37
3.2.2 Classification Performance and Scalability 38
3.2.3 Other classification features and general purposes of NASCUP 45
3.3 Discussion 51

4 Siamese CNN Based Interaction Learning 56
4.1 Methods 61
4.1.1 Notations 61
4.1.2 Input Preprocessing 62
4.1.3 Weight Shared CNN Networks 64
4.1.4 Distance and Interaction Prediction 65
4.2 Results 68
4.2.1 Experimental Setup 68
4.2.2 Effects of Hyperparameter Variation 72
4.2.3 Performance Comparison with Other Methods 74
4.2.4 Analysis on Frequent SMILES Characters 79
4.3 Discussion 83

5 Comprehensive Ensemble with CNN, RNN Based Learner 86
5.1 Background 88
5.1.1 Ensemble Learning 88
5.2 Methods 90
5.2.1 Notations 92
5.2.2 First-Level: Individual Learning 92
5.2.3 Second-Level: Meta-Learning 95
5.3 Results 95
5.3.1 Experimental Setup 95
5.3.2 Performance Comparison with Other Models 98
5.3.3 Performance Comparison on Ensemble Combination 103

5.3.4 Ensemble Effects on Class Imbalance 108
5.4 Discussion 108

6 Conclusion **111**

Bibliography **113**

Abstract in Korean **129**

List of Figures

2.1 An example of sequence logo 12
2.2 Length distribution of nucleic acid sequences 13
2.3 k-mer (1-mer, 2-mer) distribution of nucleic acid sequences 14
2.4 Examples of chemical compounds 16
2.5 Length and symbol distribution of SMILES 17
2.6 Context-tree model and context graph 23

3.1 Overall flow of NASCUP 32
3.2 Performance comparison in terms of accuracy and classification time 41
3.3 Scalability experiments on sequence-basis and family-basis expanding dataset 48
3.4 Performance changes as mislabeling noise ratio increases 50
3.5 Outlier detection with cross-entropy value 52
3.6 Synthetic sequence generation and its quality assessment with classifiers 53

4.1 Two-stage chemical analysis: feature engineering and predicting activity 57
4.2 Overview of the proposed end-to-end learning methodology 60
4.3 SMILES string length distribution 69
4.4 Dataset breakdown 70
4.5 Structure and hyperparameter optimization 73

4.6 Visualized performance comparison with respect to the method, representation, and dataset 80
4.7 Visualized performance comparison in terms of AUC, ACC, and MCC 81
4.8 Experimental validation of the commutative property 82
4.9 Effects of SMIILES input characters 84

5.1 Learning procedure of the proposed method 91
5.2 Visualized performance comparison the proposed method with other models 100
5.3 Performance comparison in terms of averaged AUC and ROC plot . 101
5.4 Model importance through the second-level learning 106
5.5 Ensemble effects on class-imbalanced datasets 109

List of Tables

3.1 Details of the dataset used in the experiments 39
3.2 Details of the expanding datasets used in the scalable experiments . 40
3.3 Classification accuracy comparison 43
3.4 Total runtime (classification time) comparison 44
3.5 Details of the runtime in the sequence-basis expanding datasets . . . 46
3.6 Details of the runtime in the family-basis expanding datasets 47
3.7 Experimental results in terms of accuracy and time on original SILVA (SS, SL) datasets 48
3.8 Accuracy comparison from candidate extraction 49

4.1 Details of the datasets used in the experiments 68
4.2 Details of experimental results on CCI900 75
4.3 Details of experimental results on CCI800 76
4.4 Details of experimental results on CCI700 77

5.1 Details of the bioassay datasets used in the experiments 96
5.2 Performance comparison the proposed method with single models . 99
5.3 Performance comparison on ensemble combination methods 104
5.4 Performance comparison of comprehensive ensemble meta-learning methods 105

Chapter 1

Introduction

A sequence can be defined as a series of related things: a series of events, movements, or symbols. Natural language, represented as a series of words or letters, is one of the most common types of sequence data; biological sequences, such as DNA and RNA are represented by a series of `A`, `C`, `G`, and `T`(`U`) symbols and proteins are represented by a series of 20 symbols. There are time-dependent sequences (time series), such as stock market data and biological signals. Additionally, drugs can be expressed by a series of symbols that represent chemical properties, such as atoms, bonds, and other structural information.

The values in a sequence are not assumed to be independent and identically distributed (i.i.d.), but rather have local or long-distance dependencies and hidden patterns of variable length. For example, in natural language, the word "`am`" is more likely to follow "`I`" than "`you`"; splice junction of a biological sequence boundary between an exon and an intron is dominantly represented by a 2-mer pattern of `GT` or `AG`; the secondary structure of transfer RNA (tRNA) is visualized as a cloverleaf—the starting region is linked with long-distance ending region; and an electrocardiogram (ECG) has a time-dependent pattern specific to each heartbeat.

The simplest approach to handle sequences is k-mer (or n-gram, in natural language applications), which refers to all the possible substrings of length k contained in a string. In the case of biological sequences, k-mer is expressed as length k of base symbols $x_1, x_2, \cdots, x_k$, where $x \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The frequency of k-mers can be used as a sequence signature. The k-mer approach is a very simple, but powerful method, as demonstrated in numerous biological applications including classification or identification in a mixture of sequences, location of motifs, characterizing CpG islands, prediction of sequence functions, and detection of sequence repetitions.

Fundamentally, the k-mer approach focuses on k size of local dependency, but by sliding-window scanning, the dependency is linked more widely. To identify the unique k-mers in a single sequence, k should be larger than $\log_4 L$, under the i.i.d. assumption, where L is the sequence length. As real sequence values are not guaranteed to be i.i.d., k should be larger than the theoretical value.

Deciding the appropriate value of k is very important. If k is too small, the model complexity decreases, such that it may be difficult to reach a high level of performance. In contrast, if k is too large, the model complexity increases exponentially, but most of the k-mers are zero resulting in sparse features. Large k value may achieve higher performance, but the model is likely to overfit.

Through the following preliminary research, we confirmed the simplicity and powerfulness of the k-mer approach and observed the empirical performance varying with the value of k.

- Sunyoung Kwon, Byunghan Lee, and Sungroh Yoon "CASPER: context-aware scheme for paired-end reads from high-throughput amplicon sequencing," *BMC bioinformatics*, vol. 15, no. 9, p. S10, September 2014,

The research exploits the frequency of k-mers in high-throughput sequencing data for paired-end read merging. When merging if two bases from the opposite side of

reads conflict (different bases at the same location), the base is decided via the k-mer frequency.

Another approach for sequence data is the flexibly changing k-mer, which changes k automatically and optimally, depending on the data. Suppose we have a sequence, x=`101010`$\cdots$`1010`, of length L. If we apply 1-mer (equivalently zeroth-order Markov model), the probability of the sequence x becomes $\frac{1}{2}^L$ by maximum likelihood (ML). However, if we apply 2-mer (equivalently first-order Markov model), the probability of the sequence x becomes $P(0|1)^{\frac{L}{2}}P(0|1)^{\frac{L}{2}} = 1$ by ML. Every symbol `0` is followed by `1`, and every `1` is followed by `0`. Based on the first-order Markov model, the probability of both $P(0|1)$ and $P(1|0)$ will be 1. Although we increase k, the probability would not exceed 1 for the given sequence x by ML. In this case, the k value of minimum complexity and maximum probability will be 2 (first-order Markov model).

We applied this flexible k-mer (variable-order Markov model) approach and confirmed its robust and generalized performance through the following research.

- Sunyoung Kwon, Gyuwan Kim, Byunghan Lee, Sungroh Yoon, and Young-Han Kim,"NASCUP: Nucleic Acid Sequence Classification by Universal Probability," in *2016 RECOMB-Seq (Massively Parallel Sequencing)*, Los Angeles, USA, April 2016 (Poster).

The research explores sequence classification against a number of families that are formed based on function or taxonomy. As mentioned the research exploits the variable-order Markov models and utilizes the notion of universal probability in the building of variable-order Markov models and classification processes. The proposed method captures the probabilistic structure of a sequence family as a compact context-tree model and uses it efficiently to test proximity and membership of a query sequence.

Using variable-order Markov models, we could build a model that would be

more compact than the fixed k-mer model. The probability for each context was given by universal probability, a mixture of Dirichlet distribution. Both the variable-order Markov model and universal probability help to avoid overfitting and to make a generalized decision. A comprehensive experimental study involving seven public databases demonstrates the advantages of the proposed method over widely used alternatives in terms of efficiency, accuracy, and scalability across all datasets considered.

Furthermore, we can detect hidden patterns or dependencies in sequences through neural networks, particularly the well-known CNNs or RNNs.

The CNN was originally applied for image and video processing, but it also has been used in sequential data, such as natural language and biological sequences with one-dimensional (1D) filters. The number of hidden parameters in the CNN is related to the filter length (similar to the k size in k-mer) and the number of filters (similar to the number of k-mers). While the k-mer consists of four discrete value, the CNN filter consists of real values. So, CNN may require a far fewer number of filters than that of the plain k-mer model.

CNNs consist of a convolution layer and pooling layer. The convolution layer can captures shift-invariant local patterns by sliding filters along a sequence. Subsequently, the local patterns are transfered to the pooling layer where they are summarized. We applied the CNN to sequences of chemical representations and confirmed the feasibility of this approach through the following research:

- Sunyoung Kwon, Ho Bae, Seonwoo, Min, and Sungroh Yoon, “End-to-end 1D-CNN based Siamese architecture for Chemical-Chemical Interaction Prediction," submitted to *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), under revision*.

- Sunyoung Kwon, and Sungroh Yoon, “DeepCCI: End-to-end Deep Learning for Chemical-Chemical Interaction Prediction," in *Proceedings of the 8th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB)*, Boston, USA, August 2017.

These studies investigate predictions based on CCI. CCI plays a major role in predicting candidate drugs, toxicities, therapeutic effects, and biological functions. CCI is typically inferred from a variety of information, however, it has not been predicted with a learning-based approach. In other recent drug analyses, deep learning has been actively used. However, in most cases, deep learning has been used primarily for classification, despite its capability for feature extraction. Thus, in the present study, we propose an end-to-end representation method for CCI named DeepCCI that employs a learning-based approach including feature extraction.

The proposed architecture is based on the Siamese network. Using weight-shared CNNs, hidden representations are extracted from a simplified molecular input line entry system (SMILES), which is a string notation representing the chemical structure. Subsequently, L1 element-size distances between the extracted hidden representations are measured. The performance of DeepCCI was compared with twelve method-fingerprint combinations. The proposed DeepCCI showed the best performance according to most of the evaluation metrics used. We confirmed that the automatically extracted features can both alleviate the efforts required for manual feature engineering, as well as improve prediction performance.

Another well-known neural network structure, RNN was originally developed for sequential data. The learning for each time point is chained sequentially. The hidden value of the current time (h_t) is determined by the hidden value of the previous time (h_{t-1}) and input of the current time (x_t), $h_t = f_w(h_{t-1}, x_t)$. By transferring the hidden values chained sequentially from beginning to end (or from end to beginning),

the RNNs can memorize locally or globally relevant information.

CNNs and RNNs are applied for the end-to-end hidden sequence-type representations of chemicals. We enhanced quantitative structure activity relationship (QSAR) prediction through the following research:

- Sunyoung Kwon and Sungroh Yoon, “Comprehensive Ensemble Learning in QSAR Prediction for Drug Discovery," in preparation

QSAR prediction identifying the relationship between chemical structure and biological activity is an invaluable process for drug discovery by which hundreds of thousands of chemicals (drugs) are narrowed down to highly reliable candidates. We applied CNN to the first hidden layer to capture local patterns and applied RNN to the next layer to understand the long-term dependencies. Operating independently, this neural-network-based classifier did not achieve particular impressive results.

We tried ensemble learning, which considers many individual classifiers together. Typically, ensemble techniques are known to achieve better prediction performance than a single classifier. We applied the proposed end-to-end CNN- and RNN-based classifier as one of the individual classifiers. In addition to the proposed methods, twelve classifiers with combinations of four types of methods and three types of chemical representations were used. For the final decision, we combined the prediction probabilities from the individual classifiers by using second-level learning (stacking). We confirmed the enhanced prediction performance of comprehensive ensemble learning. Through second-level learning, the importance weights for the individual classifiers are automatically assigned. Across all datasets, the proposed individual methods consistently showed a high level of importance.

The remainder of this dissertation is organized as follows: Chapter 2 provides a background on sequential data and analyzing methods; chapter 3 explains the biological sequence classification using the variable-order Markov model and universal

probability; chapter 4 explains the CCI prediction based on the end-to-end convolutional Siamese network; chapter 5 explains the QSAR prediction based on comprehensive ensemble learning with a CNN- and RNN-based individual classifier; and chapter 6 concludes this dissertation.

Chapter 2

Background

2.1 Sequential Data

Living beings have an origin in an ordered list of information sequence called DNA, RNA, and protein, and communicate with each other using sequential form of acoustic wave information. Diverse of signals, such as natural phenomena, biological signals, and artificial events produce sequential information depending on time flows. We are made up of or surrounded by numerous sequential information, and understanding the meaning of such sequential information is a very important task.

Sequential in sequential data means "following a particular order", according to the dictionary definition. Sequential data can be defined as a series of related things or ordered list of items: a series of events, a series of movements, and a series of symbols. Natural language as a series of words or a series of letters is one of the most common types of sequence data. Biological sequences, such as DNA and RNA is represented by a series of `A`, `C`, `G`, and `T`(`U`) symbols, and protein is represented by a series of 20 symbols. There are also time series of stock market data and biomedical signal data which are time dependent values. In addition, drugs also can be expressed

by a series of symbols which represents chemical properties, such as atoms, bonds, and other structural information.

In sequential data, an order is very important information. If the order is changed, a sequential data can be totally different things. For example, in natural language, a word and its spelled backward word mean something totally different, such as "live" and its spelled backwards "evil", "desserts" and "stressed", and "knits" and "stink", and in biological sequences, codon, which is a three consecutive nucleotides sequence, determines proteins according to the 4^3 kinds of four base symbol order.

One of the major features that makes sequential data different from existing data and makes it difficult to analyze is that it is not considered to be independent and identically distributed (i.i.d.). Rather sequential data has local or long-distance dependencies. In natural language, a word `am` is more likely to be placed following `I` than `You`. Splice junction of a biological sequence boundary between an exon and an intron is represented by 2-mer pattern of `GT` or `AG`, dominantly. The secondary structure of transfer RNA (tRNA) is visualized as the cloverleaf; the starting region is linked with long-distance ending region. In case of electrocardiogram (ECG), it also has a time-dependent specific pattern during each heartbeat.

Most common type of sequence modeling is as follows:

$$P(x_{1:n}) = \prod_{i=1}^{n} P(x_i|x_{1:i-1}) \approx \prod_{i=1}^{n} P(x_i|x_{i-k:i-1}), \tag{2.1}$$

where $x_{1:n} = x_1, x_2, \cdots, x_n$ represents n length of sequence. Since sequence has dependencies between items, it is not usually treated as totally independently, such as $\prod_{i=1}^{n} P(x_i)$ without given preceding symbols. Instead, the probability of $x_{1:n}$ is calculated by product of every symbol probability given all preceding symbols, and it is estimated by using k-mer approximation which limits the next word probability

only to the preceding $k-1$ symbols.

To retrieve inherent relationships and dependencies between items or to understand hidden patterns in sequential data, there are following challenges. Considering all the preceding symbols requires excessive computational resources and running time. An approximation with k-mer has been proposed to reduce the burdens, but it limits direct dependencies within k length of items. Memorizing long-term dependencies with vanilla RNN causes vanishing gradient problem in neural network, so LSTM, GRU, dilated-CNN are proposed to complement such problems.

Another challenge is variable length in sequential data. Unlike fixed-size input in image processing, most sequences are represented by variable length data; biological sequences within a same class or drugs have different lengths. Transformation to fixed-size feature vector, such as string kernel in biological sequences and fingerprints in drugs, have been proposed and used. However, the fixed feature vector transformation has the disadvantage of confining hidden features to predefined features. There is a way of truncating or padding to a certain fixed length, but the amount of information and efficiency in computation depends on the certain length.

Last challenge is that sequential data is big in size (or dimension) and needs very efficient computational approaches; a chemical list from STITCH [55] is more than 100 million, biological sequences in SILVA [82] is more than 1 million, and a vocabulary size is very huge in natural language processing. Compacted sequential representation algorithms or parallelized RNN approaches proposed for acceleration. Word embedding methods have been proposed for dimensionality reduction.

Sequential data have these challenges, but we try to find hidden patterns and derive meaningful results in sequential data with the understanding of the characteristics and various approaches of sequential data.

2.2 Representation of Sequential Data

2.2.1 Nucleic Acid Sequence

A nucleic acid sequence can be represented by a series of A, C, G, and T(U) as a primary structure. The nucleic acid sequence also can be transformed into secondary, tertiary, and quaternary structures. The higher level of structures can be predicted from a primary structure, which has all latent features for transforming. The primary sequence expressed by only the four symbols of A, C, G, and T(U) constitutes the basis of all living things in the world, and is a mystical set that influences all life activities. An example of a sequence is as follows:

GATGAACGCTGGCGGCGCGCCTAATACATGCAAGTCGAGCGAACGGACGAGAAG$\cdots$

from left to right in the 5' to 3' direction, A, C, G, and T(U) represent adenine, cytosine, guanine, thymine (uracil) functioning the fundamental units of genetic code. Adenine and guanine are purine bases that have a fused-ring skeletal structure, and cytosine and thymine (uracil) are pyrimidine bases that have a simple-ring structure. Each of these bases forms a purine and pyrimidine pairs in a typical double-helix DNA; adenine (A) is paired with thymine (T) or uracil (U, guanine (G) is paired with cytosine (C), called base complement.

Currently, the genetic codes (the order of four bases) are determined through sequencing technology that accelerated a huge amount of biological research and medical discovery; identifying changes in genes related with phenotypes and diseases, and discovering potential drug targets; studying evolutionary biology how different genetic codes are related to evolution; accelerating microorganism research on their types, amounts, and composition from diverse environment or organs. The next-generation (high-throughput) sequencing technology has contributed to lower costs and lower sequencing errors. Although the sequencing error rate is low, it can

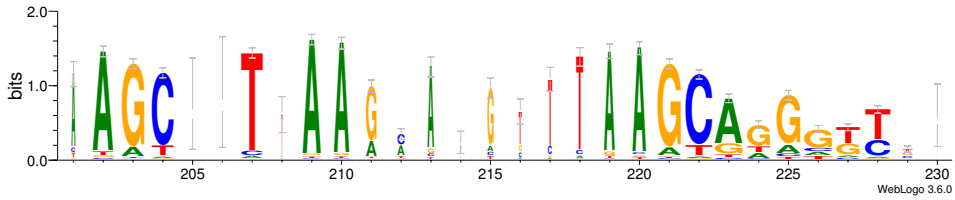

Figure 2.1: An example of sequence logo. Sequence logo is a graphical representation of sequence conservation and this logo is one of Rfam family.

cause a large number of errors due to a huge amount of sequencing reads. We have to consider the errors when we analyze nucleic acid sequencing data.

The nucleic acid sequence data are stored at a huge amount of diverse biological databases; more than 1,600 databases are categorically presented in Nucleic Acid Research Molecular Biology Database Collection. Among many databases, we focused on the family-type databases and conducted research on modeling of the family-type sequences; Rfam is a functional collection of RNA families: non-coding RNAs, structured cis-regulatory elements and self-splicing RNAs. RDP, Greengenes, and SILVA are ribosomal RNA (rRNA) of taxonomically annotated databases and the sequences are grouped to particular families depending on the taxonomic labels. SILVA consisted of two subsets; one from large subunit (LSU) 23S and 28S and the other from small subunit (SSU) 16S and 18S rRNA sequences.

Typically, functional or taxonomic family-type sequences have conserved regions, and these conserved regions play as key features for identifying sequences or distinguishing sequences from other sequences. As represented in Figure 2.1, certain regions are highly conserved (with certain symbols dominant) without little mutation; location 207 is `T` dominant and location 210 is `A` dominant. We inspected the basic characteristics of these family-type sequences (rfam, silva, and greengenes). As represented in Figure 2.2, the distribution of the sequence length varies depending on the data type; the lengths of Rfam sequences are mostly shorter than 500, those of

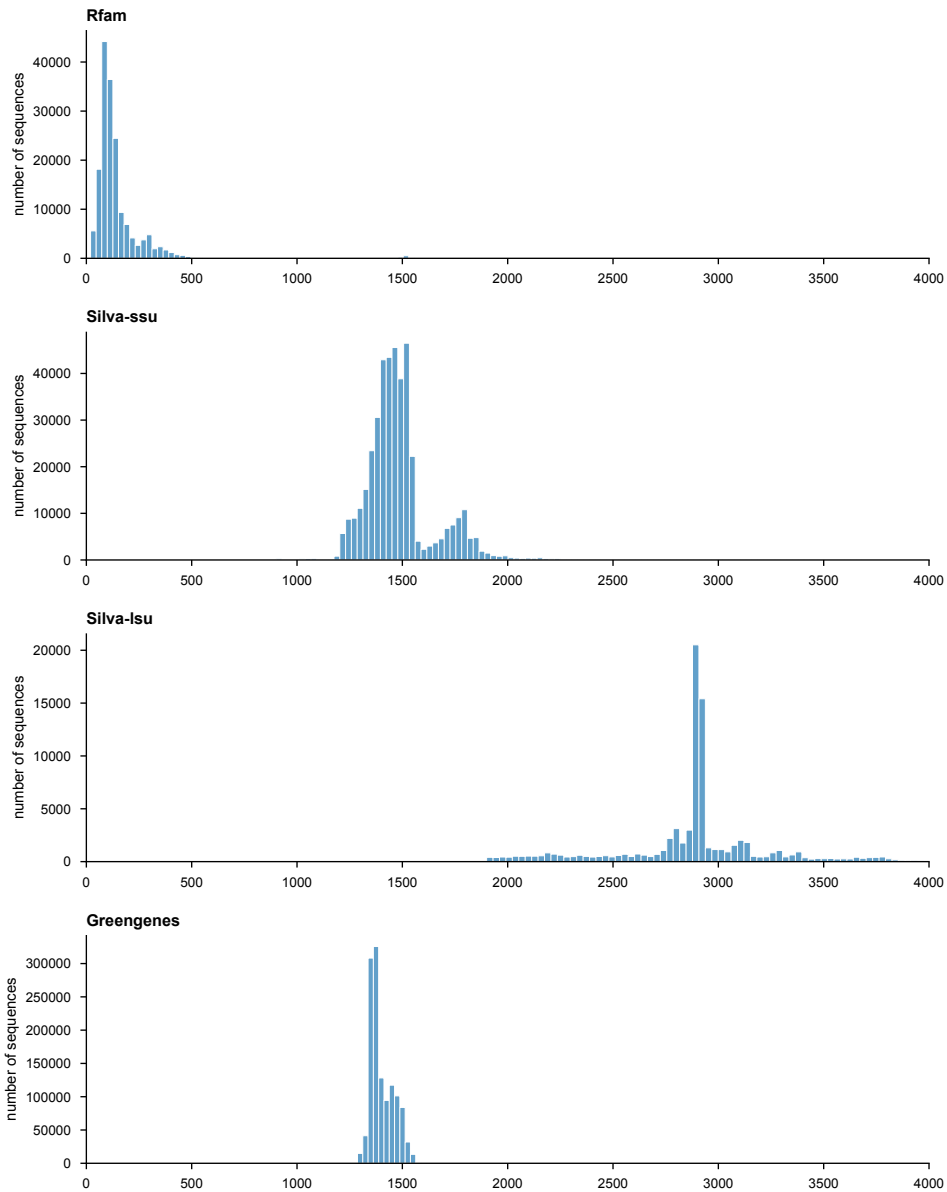

Figure 2.2: Length distribution of nucleic acid sequences: Rfam (v11.0) [35], silva (ssu, lsu) (v132) [82] , greengenes (v13.5) [23].

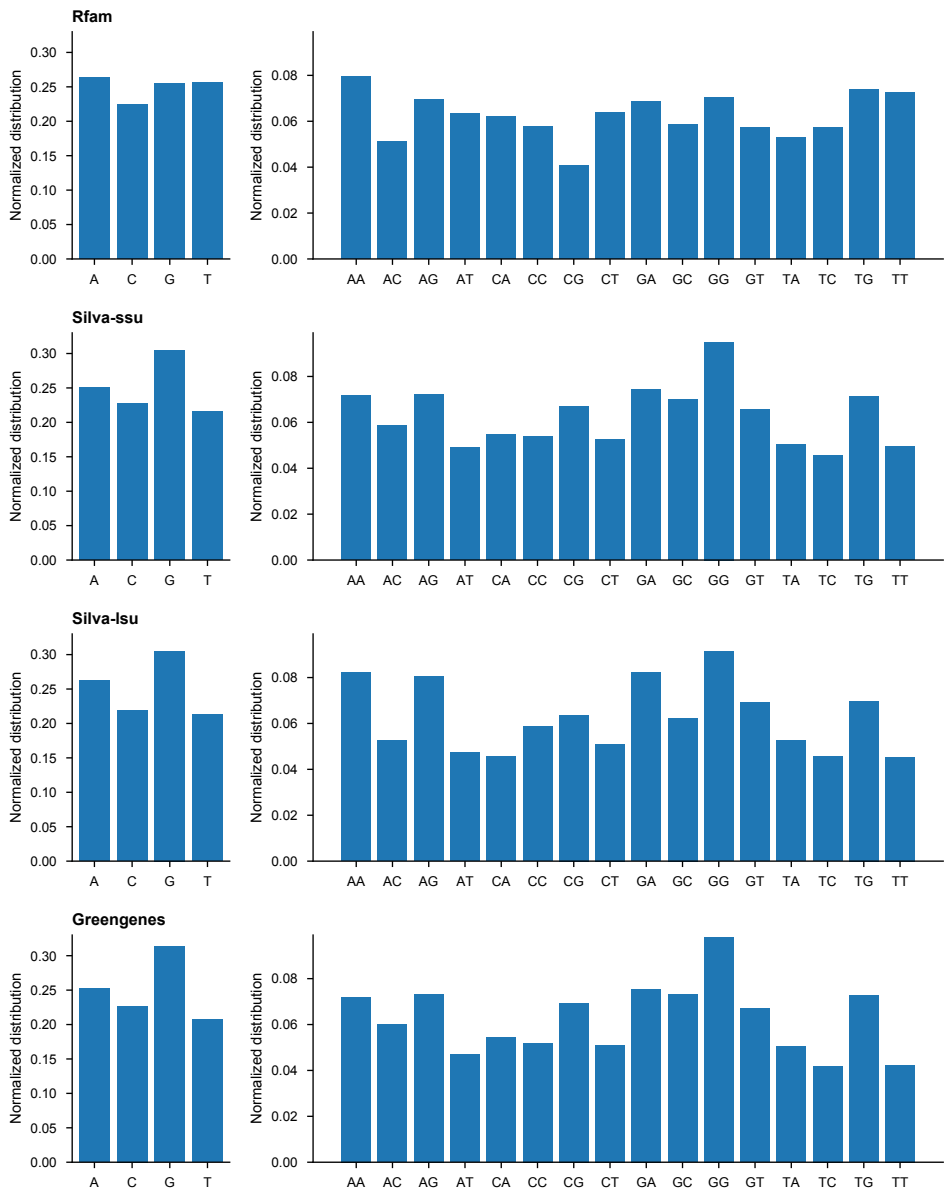

Figure 2.3: k-mer (1-mer, 2-mer) distribution of nucleic acid sequences: Rfam (v11.0) [35], silva (ssu, lsu) (v132) [82] , greengenes (v13.5) [23].

Silva-ssu and Greengenes are between 1,000 and 2,000, and the sequences of Silva-lsu are around 3,000 in length. As represented in Figure2.3, each symbol of `A`, `C`, `G`, and `T` is evenly distributed throughout, although there is a slight tendency depending on the data; each symbol has a ratio of at least 0.2 and each 2-mer symbol has a ratio of at least 0.04 relatively evenly; the three ribosomal RNA datasets (Silva-ssu/lsu, Greengenes) shows similar symbol distribution distinguishing from Rfam dataset.

2.2.2 Chemical Compound

Chemical compounds have been expressed in various representation types, such as molecular formula, 2D structure, 3D conformer, InChI, SMILES, and fingerprints. The representation of chemical compound not only describing the chemical itself, but also is used for *in silico* virtual screening or drug discovery. Therefore, the representation of chemical compound should contain as many chemical properties as possible. Traditionally, diverse fingerprints have been used for a long time, and there have been attempts using SMILES with deep learning.

SMILES

The simplified molecular input line entry system (SMILES) [111] is a line notation for describing the structure of a chemical compound with ASCII characters. The original SMILES was initiated by David Weininger in the 1980s. It has been modified and extended notably by Daylight Chemical Information System. Generally, SMILES is considered a human-readable structural representation of chemical compound.

As a graphical approach, SMILES is obtained by expressing encountered nodes and bonds in a chemical graph by depth-first traversal. Structural information of chemical compound, such as atoms (*e.g.*, carbon, nitrogen, and oxygen), bonds (*e.g.*, single, double, and triple bonds), rings (*e.g.*, open ring, close ring, and ring number),

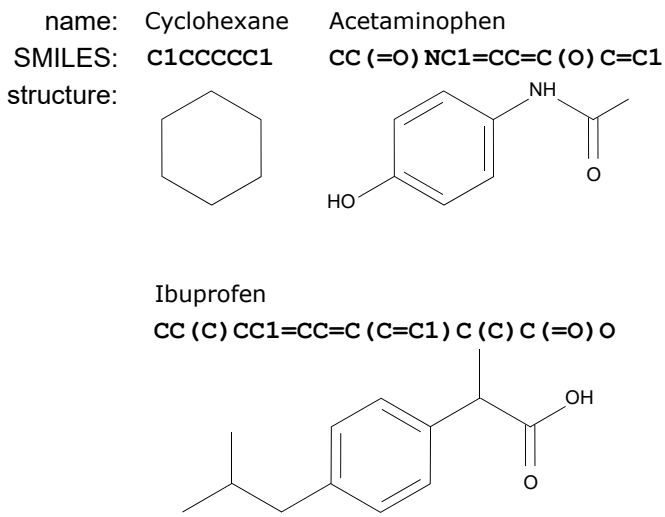

Figure 2.4: Examples of chemical compounds: 2D structure, name, and SMILES of cyclohexane, acetaminophen, and ibuprofen.

aromaticitiy, and branching can be represented with SMILES. Generation of SMILES is as follows:

- First, all hydrogen atoms are removed.
- Second, each cycle is broken and a numeric label indicating the cycle is added to the broken cycle opening and closing atoms.
- Third, parentheses are used for branching off the main backbone.

Cyclohexane, acetaminophen, and ibuprofen are expressed as SMILES of `C1CCCC1`, `CC(=O)NC1=CC=C(O)C=C1`, and `CC(C)CC1=CC=C(C=C1)C(C)C(=O)O`, respectively, as shown in Figure 2.4, (`C` for carbon, `O` for oxygen, = for double bonds, and `1` for the first ring). Acetaminophen, as shown in Figure 2.4, is expressed starting from the right upper side to the left down as backbone, branching double bond `O` as `(=O)`, opening the cycle as `C1`, branching `HO` as `(O)`, closing the cycle as double bond as `=C1`. However, the same acetaminophen can be expressed with multiple

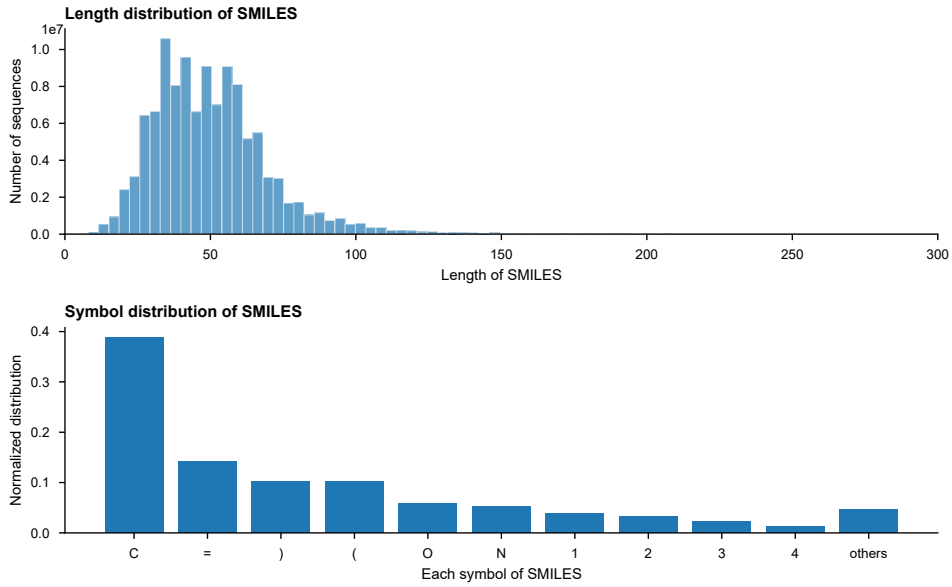

Figure 2.5: Length and symbol distribution of SMILES. Top figure is for length distribution of SMILES and bottom figure is for symbol distribution of SMILES.

SMILES notations, *e.g.*, `CC(=O)NC1=CC=C(C=C1)O`, `CC(=O)NC1CCC(O)CC1`, depending on the starting point, broken point of a cycle, and choice of main backbone in a chemical graph. This multiple expression can be used as a data augmentation technique. The augmented SMILES dataset can be 130 times bigger than only one canonical SMILES dataset and showed better performance [7].

We analyzed length and symbol distribution of SMILES, as shown in Figure 2.5. The set of chemical information (chemical index, name, molecular weight, SMILES) was downloaded from STITCH database [55], and it contains more than one million chemical compound information. Most of SMILES is shorter than 100 in length, median 48 and mean 50 lengths, and it is comparatively shorter than nucleic acid sequences. Unlike nucleic acid sequence (`A`, `C`, `G`, and `T` are evenly distributed relatively), the symbol `C`, representing carbon, accounts for 38% dominantly in SMILES. Nearly half of the symbols are represented with `C`, and more than 95% of symbols in SMILES are expressed with just 10 characters; carbon, double bond, branching,

oxygen, nitrogen, cycle number 1 to 4. The rest of symbols in SMILES only accounts for about 4%.

As a one-dimensional representation of the chemical structure, SMILES can be converted into a two- or three-dimensional chemical structure, which means that it contains sufficient structure information to transform into higher dimensions. SMILES is also used for drug generation [33, 91] and for compound similarity [78]. It is generally known that structure is closely related to function [87, 96]. The QSAR prediction, for example, analyzes the relationship between the chemical structure of a compound and its biological activity. As an expression of structure information, SMILES can be a means of clarifying the function of a chemical compound. For these reasons, we used SMILES as a front-end input format for representing a chemical compound.

Molecular Fingerprints

Traditionally, molecular fingerprints have been used for a long time in drug analyzes due to their convenience and speed. The fingerprints can be considered as extracted features, and therefore they can be easily used as inputs for machine learning approaches. Fingerprints are represented as a bit string, mostly binary vector. There are vast amounts of fingerprints, and they are categorized depending on how the chemical compounds are transformed into fingerprints [11]. We represents three major fingerprint types as follows:

- Substructure keys-based fingerprints

  Bit values are set depending on the presence of certain substructures from a predefined list of structural features. The predefined features should contain important substructure and cover most of the substructure of chemical compounds. The dimension of this fingerprints is determined by the size of the predefined features. MACCS (166 dimension) and PubChem (881 dimension)

are the most prominent fingerprints.

- Topological or path-based fingerprints

  Fingerprints are formed depending on the path up to a maximum length and hashing the paths. The path extension proceeds linearly from a single starting atom to every atom in a molecular graph. Daylight fingerprint is the most commonly used.

- Circular fingerprints

  Circular fingerprints are similar to the path-based fingerprints, but the path extension (building fragments) proceeds based on a radius. These types of fingerprints are used for similarity measure between chemicals and structure-activity modeling. The Extended-Connectivity Fingerprints (ECFPs) is the most widely used circular fingerprint. Depending on the diameter, ECFP4 which uses a diameter of 4 and ECFP6 which uses a diameter of 6 are commonly used.

2.3 Preprocessing of Sequential Data

The raw primary structure of nucleic acid sequence or SMILES are successions of symbols in variable lengths, the strings themselves are difficult to handle with conventional machine learning approaches. In most case, we have to convert a sequence into a numeric value in order to handle them in computational approaches. We explain the preprocessing procedure of sequential data as follows.

- Tokenization

  We could handle the nucleic acid sequences and SMILES as a single character-level or substring-level (k-mer). Generally, in case of nucleic acid sequences, character-level preprocessing is used for deep learning, and substring-level

preprocessing is used for conventional learning or information theoretic approaches. For natural language processing, predictions are often being made based on word-level tokenization.

- Limiting input dimension

  `A`, `C`, `G`, and `T` in nucleic acid sequences appear relatively uniformly, but k-mers become sparse as k increases. SMILES consists of 64 characters, but most of them (54 characters) are used very rarely, represented only under 5%. Generally, input dimension depends on the character size (vocabulary size in case of word-level tokenizing). Input dimension affects the number of learning parameters, processing time, model capacity, and quality of learning. Therefore, we need to consider limiting the input dimension to the most common tokens.

- Truncating or padding the sequences

  The most notable feature of sequential data is that the length is variable. In case of count features by using k-mers or molecular fingerprints, the size of the feature vector (input shape) is determined regardless of the length of the sequence. However, if we handle the sequences as raw as possible, such as character-level preprocessing, the length of input vector may vary depending on the sequential data. Sequential data may need to be truncated or padded to a certain length for afterward analyzing or learning procedure.

- Encoding

  For numerical changes, we could use one-hot encoding or word embedding techniques. One-hot encoding has an input dimension as the vocabulary size, and it has only a single 1 value (only a corresponding token) and all others 0 values. As the vocabulary size increases, the input dimension increases and becomes sparse. An example of one-hot encoding of DNA string (`ACCGTA`) is

as follows:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \texttt{A} \\ \texttt{C} \\ \texttt{C} \\ \texttt{G} \\ \texttt{T} \\ \texttt{A} \end{bmatrix} \overset{\{\texttt{A},\texttt{C},\texttt{G},\texttt{T}\}}{=} \begin{bmatrix} [1,0,0,0] \\ [0,1,0,0] \\ [0,1,0,0] \\ [0,0,1,0] \\ [0,0,0,1] \\ [1,0,0,0] \end{bmatrix}$$

Word-embedding, *e.g.*, word2vec, GloVe, expresses semantically similar tokens as similar vectors, similar geometric location. Input dimension of embedding is typically much smaller than original vocabulary size, and becomes dense. Embedding has two functions, reducing the input dimension and learning the tokens. In case of natural language processing, embedding is almost always used. However, small dimension sequences, such as nucleic acid sequences both of them are used.

2.4 Methods for Processed Sequential Data

2.4.1 Information Theoretic Approaches

K-mer

K-mers are all the possible k length of substrings that are contained in a sequence (in NLP, n-grams which are n number of continuous items in a document). Input dimension is determined by the token size and k, *e.g.*, 4^k in nucleic acid sequences; if k is 2, input dimension is $4^2 = 16$, consisting of AA, AC, AG, AT, $\cdots$, TT. The frequency as counting feature or existence of the k-mers can be used as a signature

of sequences. The k-mer approach has been used in a great amount of bioinformatic applications, such as classification/identification taxonomy in a mixture of sequences, prediction of functions of sequences, characterizing CpG island, features for denoiser for NGS, clustering, composition analysis for metagenomics. The k-mer approach is the simplest, but it has shown very powerful performance with scalability (space-time tradeoff). Based on k-mer, conditional probabilistic approaches of context-tree models, fixed-order Markov models, and variable-order Markov models are used.

Context and context-tree models (CTM)

A context is a given state or conditioned symbols and is used for the probability of the following symbol. In relation with k-mers models, contexts are represented by the length of depth, $d = k - 1$, and all possible 4^d states, *e.g.*, if 3-mer is `AGT`, context is `AG` and the last symbol `T` is treated as the following symbol. A context graph consists of nodes representing all potential contexts $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}, \star\}^d$, where $\star$ denotes "don't care". Leaf nodes consist of $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^d$ and root node consists of $\{\star\}^d$. Any subtree of this context graph is called a context-tree model (Figure 2.6). Context-tree model consisting of all leaf nodes, $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^d$, is referred to as the k-mer model (or d-th order Markov model). When a given context-tree model only has contexts of the form $\{\star\}^j \times \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}^{d-j}$ for some j, it is referred to as a variable-order Markov model.

Variable-order Markov models

The simplest probability models considered in bioinformatics and other scientific disciplines are k-mer models (see, for example, [17, 77, 86, 116]) or equivalently Markov models. Albeit straightforward to theorize and efficient to implement, the generic class of d-th order Markov processes suffer from the exponentially increasing large number of *contexts*; most of the contexts do not appear in a sequence, as

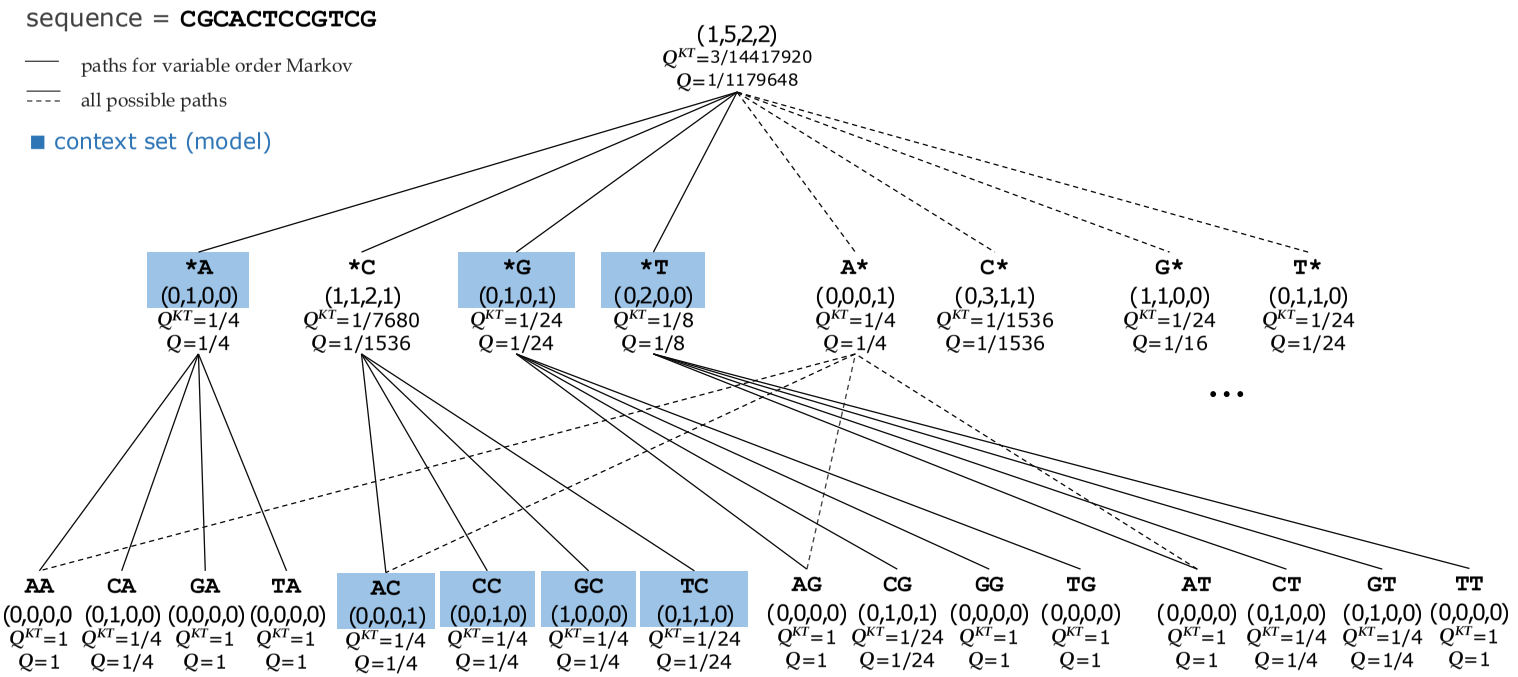

Figure 2.6: Context-tree model and context graph. A context-tree model is any subtree of context graph, consisting of all possible contexts, $\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T},\star\}^d$, where $\star$ denotes "don't care". Leaf nodes consist of $\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}^d$ and root node consists of $\{\star\}^d$. Parent node has one more $\star$ replacing the $\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}$ in child nodes, *e.g.*, $\star\mathtt{C}$ is a parent node of child-nodes of $\mathtt{AC}$, $\mathtt{CC}$, $\mathtt{GC}$, and $\mathtt{TC}$. Root node $\star\star$ is a parent node of child-nodes of $\star\mathtt{A}$, $\star\mathtt{C}$, $\star\mathtt{G}$, and $\star\mathtt{T}$. Solid line represents paths for variable-order Markov models, consisting contexts of $\{\star\}^j \times \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}^{d-j}$ for some j.

d increases; the small number of occurrences of each context may not contribute to any statistically meaningful structure and instead cause unnecessary overfitting. In contrast to the fixed d-th order Markov models, variable-order Markov models use a flexible length of context from zero to d-th order that ultimately results in compacted set of and meaningful contexts. The meaningful contexts can be determined automatically in the process of context-tree maximization.

2.4.2 Convectional Machine Learning Approaches

The raw primary structure of nucleic acid sequence itself is a succession of symbols in a variable length, it is difficult to handle with conventional machine learning approaches. A string kernel, as a kernel function operates on sequences, measures the similarity between sequences, and it translates the variable length sequences into fixed-length and real-valued feature vectors. Count vectors or frequencies of k-mers can be one of a string kernel function. Prediction by using string kernel with conventional machine learning approaches such as support vector machine (SVM), random forest (RF), and adaptive boosting (AdaBoost) can be performed. Span filtering [4] and biological protein sequence classification [61] used string kernel with SVM.

2.4.3 Deep Neural Network Based Approaches

Deep learning has been applied and produced outstanding results in various fields including image processing, natural language processing, speech recognition, bioinformatics, drug design, and game programs. Multiple layers with nonlinear units enable the neural network models to learn complex problems that were previously difficult. The artificial neuron basic unit consists of summing junction and activation function, $\sigma(b + \sum_i w_i x_i)$. Input data is transformed into a more abstract and composite representation while moving to upper layers.

For modeling of sequential problems, deep neural networks can be used; modeling with extracted features by replacing the conventional machine learning; modeling of end-to-end neural network with raw sequential data. According to the form of hidden features that reside in sequential data, various deep neural architectures can be used, such as feed forward neural networks (FFNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

Fully Connected Feedforward Neural Networks

Fully connected feedforward neural network (FFNN) is one of the most typical deep neural network architecture. Data flows from the input layer through the hidden layers and to the output layer in a feed-forward way, without any recurrent loops or cycles. All neurons in a layer are directly connected with the neurons of the subsequent layer fully; there are nm number of connections (weights) between n number of neurons and m number of neurons of subsequent layer.

FFNN is commonly used for extracted or manually crafted features as back-end classifier; FFNN with molecular fingerprints for quantitative structural activity relationship (QSAR) prediction [20, 84] and chemical-protein interaction [103] has been used as classifier; many neural network models end with FFNN for final prediction from intermediate hidden features; output features from a string kernel, count vector or frequencies from k-mers can be used as input for FFNN.

Convolutional Neural Networks

Convolutional neural networks (CNNs) [53] have been commonly applied in visual imagery [53, 57, 76] based on weight-shared architecture and convolution process. Designed to analyze shift invariant spatial information, CNNs consist of convolution layers and pooling layers. At each convolution layer, learnable feature maps called

filters scan the sub-regions across the input. Filters enable CNNs to discover locally correlated motifs regardless of their locations. Then, each pooling layer summarizes sub-regions and aggregates local features into more global features.

For image processing, the neural networks learn the filters without prior knowledge and human efforts. The number of learnable parameters depends on the number of filters and the size of filters, and the number of parameters is much smaller than that of FFNN.

CNNs have shown outstanding performance in one-dimensional (1D) biological sequences [2, 119] and linguistic sentences [44, 47], as well as in two-dimensional image processing. Each filter is convolved across the sequence on sequential data instead of the width and height on image data, and so is the pooling operation. One-hot encoded or embedded raw sequential data is fed into input layer, and weights are learned for final decision. Sequential patterns or motifs which have local dependencies in nucleic acid sequences and SMILES can be identified by CNNs.

Recurrent Neural Networks

Recurrent neural network (RNN) is a representative model designed for sequential data, processing one symbol at a time. RNNs have recurrent connection between nodes along a sequence with a directed one way, unlike feedforward neural networks. Therefore, hidden nodes affects all subsequent hidden nodes potentially, RNNs can capture long-term dependencies hidden in sequential data; hidden node at time t affected by the hidden node at time $t-1$, $h_t = f_w(h_{t-1}, x_t)$. With these architectural characteristics, RNNs are suited to handle sequential problems, such as handwriting recognition, machine translation, language modeling, image captioning, biological sequence classification, and speech recognition.

Vanilla RNNs suffer from gradient vanishing or exploding problems, as time t

increases. To overcome these shortcomings, long short-term memorys (LSTMs) and gated recurrent units (GRUs) have been proposed by adding gated state or memory, and these new approaches improved the performances of sequential problems.

RNNs have various architectures depending on the configuration of input nodes, hidden nodes, and output nodes; one-to-one for image classification, one-to-many for image captioning, many-to-one for sentimental analysis, many-to-many delayed for machine translation, and many-to-many instantly for video classification are generally used. For sequence classification, many-to-one (for final decision) or many-to-many (as intermediate layers) RNN architectures can be used.

Nucleic acid sequences or SMILES can be used as input for RNNs, and long term dependencies in a sequence can be understood. The mixture architecture of RNN and CNN (RNN layer stacked on top of 1D-CNN layer) can be used for sequential problems.

Chapter 3

Variable-order Markov Model Based Sequence Classifier

Biological sequence classification plays a key role in various bioinformatics pipelines by revealing the proximity and membership of a sequence to known sequence families [36, 68, 101, 107, 117]. Expedited by new sequencing technologies, the amount of biological information is growing rapidly, making the need for having an effective classification method more important than ever. Existing methods, however, often suffer from common limitations in time and space efficiency of training and testing, classification accuracy, or scalability.

Common approaches for sequence classification can be broadly grouped into two categories—alignment-based and model-based ones. In alignment-based approaches, the similarity between a query sequence and the sequences in a database is measured by sequence-to-sequence alignment, which can be then used to determine the class of the query sequence. Widely used alignment tools include BLAST [3], FASTA [80], BLAT [46], SOAP [63], USEARCH [28], UBLAST [28], and caBLAST [67]. As the size of the database grows, the time demand for database-wide classification often

becomes prohibitively costly. To address this challenge, various speedup techniques have been proposed (*e.g.*, seed-and-extend [3, 80], in-memory processing [46, 63], word-counting based prioritizing [28], and database compression [67]), which, however, typically sacrifice accuracy for efficiency.

Several types of model-based methods exist. First, probabilistic model-based approaches derive a model from the sequences in a family and perform the sequence-to-model comparison in order to classify query sequences. Classification techniques based on hidden Markov models (*e.g.*, HMMER [26, 27, 112] and SAM-T97 [45]), despite their popularity, often suffer from excessive running time caused by iterative training of the model and preprocessing by multiple sequence alignment (MSA). Techniques based on the interpolated context model (ICM) [89] (*e.g.*, GLIMMER [21] and Phymm [9]) show improved time efficiency but often fail to achieve the desired accuracy due to the greedy nature of their search strategy.

Second, k-mer based approaches, such as RDP [17], NBC [86], Kraken [116], and CLARK [77], extract k-mer counts from the data and then perform classification using the acquired k-mer spectra. On the theoretical side, these methods involve somewhat arbitrary metrics for comparison of k-mer spectra, such as the Euclidean distance. On the practical side, the need for storing and processing k-mer data often limits the scalability and efficiency of many k-mer based methods.

Compression-based approaches [51, 66] constitute the third, and the least popular, type of model/ feature-based methods. Roughly speaking, these approaches utilize the marginal description complexity of a query sequence when it is compressed as a sequel to database sequences; if the query sequence belongs to the family, then it would be compressed efficiently. The lack of a separate modeling process and the difficulty of compressing all sequences in a family limit the wide adoption of these techniques, not to mention their relatively poor accuracy.

To address the limitations of existing sequence classification methods, we propose the nucleic acid sequence classification by universal probability (NASCUP) method. NASCUP uses a model-based approach with characteristics related to all three types of model-based approaches—based on probabilistic models, k-mers, and compression—but with the following advantages over the existing methods:

- **Versus alignment based:** NASCUP does not rely on any sequence-to-sequence comparison, and the classification time of NASCUP is independent of the number of sequences per family, which enables a significant speedup. For gigabase-scale databases, such as Greengenes [23] and SILVA [82], many alignment-based methods would take days or even weeks for database-wide classification, whereas the time demand of NASCUP is orders-of-magnitude lower.

- **Versus k-mer based:** NASCUP utilizes a hierarchical model called *the context-tree model*, which is constructed from the "contexts" of training sequences using their k-mer information. In addition to alleviating the efficiency problem of existing k-mer based approaches (sparse vs. dense trees) in representing the probabilistic structure of a sequence, these contexts also capture the textual information of the sequence by only keeping segments of statistical importance.

- **Versus compression based:** In deriving a good context-tree model, NASCUP employs the notion of minimax universal probability measure used in information theory [16, 52, 85]. This allows for efficient and effective modeling and classification of multiple sequences in two stages, improving significantly upon the performance and scalability of compression based approaches even though NASCUP shares much of their philosophical foundations.

- **Versus probabilistic model based:** NASCUP runs faster and returns more accurate classification results than HMM-based and ICM-based alternatives.

Furthermore, NASCUP does not need any MSA preprocessing, which allows additional performance boosts over the HMM approaches that normally require MSA preprocessing. Compared with ICM, which also utilizes context trees, NASCUP implicitly enumerates all context tree models of a given depth and traverses them so that the most probable CTM can be found (cf. [75, 115]). This principled approach of tree building in NASCUP pays off with its much improved accuracy. Even so, the running time of NASCUP typically remains lower than that of ICM.

We have performed a set of comprehensive experiments that test NASCUP with a variety of real sequence datasets for functional non-coding RNA classification, microbial taxonomy classification. To see the feasibility as a preprocessor and robustness in noisy datasets, and to identify the advantage as a generative model that can be used for general purposes, we performed additional experiments. Our experimental results confirm the effectiveness of NASCUP and its advantages over existing state-of-the-art in terms of time and space efficiency and classification accuracy. As a generative model, moreover, NASCUP showed robust performance across diverse and noisy datasets, and it also can be used for general purposes such as outlier detection and synthetic sequence generation.

3.1 Methods

NASCUP aims to extract the structure of each nucleotide sequence family by relative frequencies of the symbols `A`,`C`,`G`,`T`(`U`) in the constituent sequences (model building) and utilize this structural information to assign the query sequence to the best fit family (classification) (Figure 3.1). Unlike the simplistic k-mer approach, however, NASCUP leverages the notion of *universal probability* in both model building and

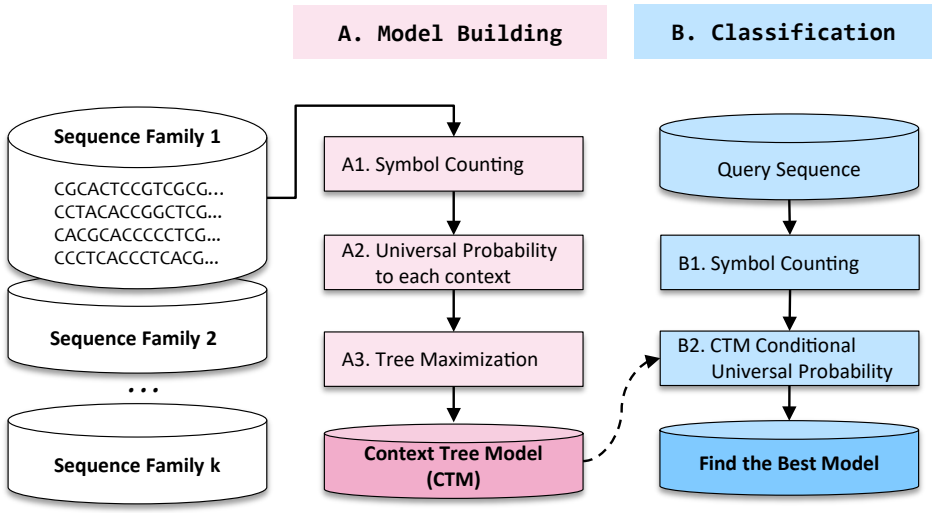

Figure 3.1: Overall flow of NASCUP. The proposed NASCUP methodology consists of **(A)** Model-building and **(B)** Classification pipelines leveraged by the notion of universal probability in step A2 and step B2. Context-tree model per sequence family was built with contexts of highest universal probability and the class of query sequence was determined by the highest universal probability given context tree model.

classification pipelines.

Universal probability

The theory of universal probability has been developed in ergodic theory, information theory, and online learning. Roughly speaking, a universal probability distribution Q approximates all distributions P in a given class of distributions uniformly well and thus serves as a simultaneously good proxy. The existence and construction of such universal probability distributions are well-known for bounded parametric families [16, 85]. The famous Lempel–Ziv incremental parsing algorithm [122] used in data compression can be recast as an assignment of probability that is universal with respect to the class of all finite-alphabet stationary ergodic processes. NASCUP uses Krichevsky–Trofimov universal probability Q that can approximate quaternary stationary Markov process P with very reasonable lower-order terms.

3.1.1 Model building

In the model building stage, NASCUP finds the best CTM for each family of sequences by finding the CTM that maximizes the *universal probability* with respect to all distributions under the CTM. This maximum likelihood model selection approach is well justified by statistics and information theory, with our variation being the use of the universal probability as a minimax optimal representative for a class of potential distributions. The universal probability maximization can be performed recursively, as demonstrated earlier by [75, 115] for data compression applications. Figure 3.1 represents overall flow of NASCUP.

Step 1: Symbol counting for each context. NASCUP initially counts the nucleotide symbols A,C,G,T that follow each of length-d contexts in $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^d$ for all sequences in the family. The counts of the symbols at each context $s \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, *\}^d$ is denoted as n_i, $i \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, represented with 4-dimensional count vector, $\mathbf{n}(s) = (n_{\mathtt{A}}, n_{\mathtt{C}}, n_{\mathtt{G}}, n_{\mathtt{T}})$. These counts are then merged bottom–up from the leaf nodes to the root iteratively, *e.g.*,

$$\begin{aligned}\mathbf{n}(\star\star) &= \mathbf{n}(\star\mathtt{A}) + \mathbf{n}(\star\mathtt{C}) + \mathbf{n}(\star\mathtt{G}) + \mathbf{n}(\star\mathtt{T}) \\ &= \mathbf{n}(\mathtt{A}\star) + \mathbf{n}(\mathtt{C}\star) + \mathbf{n}(\mathtt{G}\star) + \mathbf{n}(\mathtt{T}\star).\end{aligned}$$

Step 2: Universal probability assignment to each context. The Krichevsky–Trofimov (KT) [52] probability distribution is asymptotically minimax universal with respect to i.i.d. processes with unknown parameters [16, 85] and is used widely in universal data compression [18]. Hence, given $\mathbf{n} = \mathbf{n}(s)$ for each context s, NASCUP computes the Krichevski–Trofimov (KT) [52] probability

$$Q^{\mathrm{KT}}(s) = Q^{\mathrm{KT}}(n_{\mathtt{A}}, n_{\mathtt{C}}, n_{\mathtt{G}}, n_{\mathtt{T}}) = \frac{\prod_{i\in\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \prod_{j_i=1}^{n_i} (j_i - 1/2)}{\prod_{j=1}^{n_{\mathtt{A}}+n_{\mathtt{C}}+n_{\mathtt{G}}+n_{\mathtt{T}}} (j+1)}, \tag{3.1}$$

which can be viewed as a Dirichlet mixture of independent and identically distributed (i.i.d.) random processes over the quaternary alphabet.

Step 3: Recursive context tree maximization. For a context tree model with a set S of contexts, (3.1) results in the probability of the entire sequence (or sequence family) given by

$$\prod_{s \in S} Q^{\mathrm{KT}}(s). \tag{3.2}$$

This *conditional* KT probability distribution continues to be minimax universal (see, for example, [85, 115]), if computed with the correct context tree model. NASCUP finds the best context tree model, or equivalently, S that maximizes (3.2) recursively by starting with the root node in the context graph in a depth-first manner and computing for each context s

$$Q(s) = \max\{Q^{\mathrm{KT}}(s), \prod_{\text{child } s' \text{ in each set}} Q(s')\}, \tag{3.3}$$

where s' is the children sets of s. If s is a leaf (at depth d), then $Q(s) = Q^{\mathrm{KT}}(s)$. As an example, for $d = 2$,

$$\begin{aligned} Q(\star\star) &= \max\{Q^{\mathrm{KT}}(\star\star),\, Q(\star\mathtt{A})Q(\star\mathtt{C})Q(\star\mathtt{G})Q(\star\mathtt{T})\}, \\ Q(\star\mathtt{A}) &= \max\{Q^{\mathrm{KT}}(\star\mathtt{A}),\, Q^{\mathrm{KT}}(\mathtt{AA})Q^{\mathrm{KT}}(\mathtt{CA})Q^{\mathrm{KT}}(\mathtt{GA})Q^{\mathrm{KT}}(\mathtt{TA})\}. \end{aligned}$$

Children set that does not attain the maximum is pruned. In case of a tie between a parent context and its children set, we choose the parent for the sparsity of the resulting model. Upon completion of this maximizing and pruning step, the remaining nodes of the graph is either a leaf or a non-leaf with a single non-overlapping set of four children. This tree and the count vectors of its nodes (contexts) form the context-tree model (CTM) for the given sequence family.

3.1.2 Classification

The notion of universal probability is utilized in the classification stage as well. For simplicity, consider two families of sequences with true distributions P_1 and P_2, and two universal probability distributions Q_1 and Q_2, each corresponding to the best CTM for a given family selected in the model building stage. Let $\mathbf{y}_1$ and $\mathbf{y}_2$ be the sequences (collectively) in each family and $\mathbf{x}$ be the query sequence we wish to classify. Were P_1 and P_2 known, then the optimal Bayes classifier would compare

$$\log P_j(\mathbf{x}), \quad j = 1, 2,$$

and choose the one with the maximum likelihood—or compute their ratio (log-likelihood ratio) and classify according to some threshold. If we assume that the correct CTMs were found, then

$$\log Q(\mathbf{x}|\mathbf{y}_j) \approx \log P_j(\mathbf{x}|\mathbf{y}_j) \approx \log P_j(\mathbf{x}), \quad j = 1, 2, \tag{3.4}$$

where the conditional probabilities denote those of the concatenated sequence, say, $Q(\mathbf{y}_j, \mathbf{x})$, divided by the probability of the sequence, say, $Q(\mathbf{y}_j)$. Here the first approximation is justified by the universality of Q and the second approximation is justified by the Markovity of P_j. Hence, approximately we can perform Bayes classification or any log-likelihood ratio test even without knowing the true probability distributions. The idea of using universal probability in classification traces back to [37, 121] and NASCUP extends it to CTMs and associated minimax universal probability distributions. Note that the measure, $-\log Q(\mathbf{x}|\mathbf{y}_j)$, can be interpreted as the length of the optimal source code for the distribution Q [18]. In this sense, NASCUP can be viewed as a refinement of compression-based classification meth-

ods [51, 66] that uses the optimal source code using universal probability.

Assume that there are K known families of sequences in the database. Given an unlabeled input sequence $\mathbf{x}$, NASCUP computes the membership scores of the sequence for the K context-tree models in three steps.

Step 1. Symbol counting for each context. As in the modeling stage, NASCUP generates count vectors, $\mathbf{m}(s) = (m_{\mathrm{A}}, m_{\mathrm{C}}, m_{\mathrm{G}}, m_{\mathrm{T}})$, for query sequence.

Step 2. Universal probability computation for each family. Fix a context-tree model M characterized by a context tree (a set S of contexts) and the count vectors $\mathbf{n}(s)$ for $s \in S$. Following (3.4), NASCUP computes the conditional probability of $\mathbf{x}$ given M as

$$Q(\mathbf{x}|M) = \prod_{s \in S} Q(\mathbf{m}(s)|\mathbf{n}(s)), \tag{3.5}$$

which is computed again according to the KT probability assignment in (3.1), which has a simple Bayesian update for the conditional probability computation:

$$Q(\mathbf{m}|\mathbf{n}) = \frac{Q^{\mathrm{KT}}(\mathbf{m}+\mathbf{n})}{Q^{\mathrm{KT}}(\mathbf{n})}. \tag{3.6}$$

Step 3. Finding the best model. NASCUP compares the conditional probabilities in (3.6) over all K families and selects the family with the largest probability.

3.1.3 Outlier detection in a family of sequences

Regardless of the existence of outliers, NASCUP tries to build a context-tree model that can represent the all sequences in a family, including the outliers. The generated model reflects the nature of true sequences that account for the majority, rather than the outlier features because the number of outliers is much smaller than the number of true sequences. To detect outliers within a sequence family, we used the all sequences in a family as training and also as query sequences. With the cross-entropy value of

a query sequence given the model, averaged in sequence length, we can determine the degree of conformance to the model of a query sequence: the smaller the cross-entropy value, the better conformance to the model, and the larger the value, the greater difference from the model, indicating the greater likelihood of outliers. If the cross-entropy values of some sequences are distinctly large compared to the other sequences, we can predict them as outliers (Figure 3.5).

3.1.4 Synthetic sequence generation from learned CTM

With contexts and their count vectors, represented in a context-tree model (CTM), we can generate synthetic sequences. The first context (depth-length of symbols) is designated as the most dominant starting context from the existing sequences and then the following symbol from a given context is predicted sequentially by sliding window. The following symbol is determined (sampled) by the Dirichlet distribution from the count vector, $(n_{\mathtt{A}}, n_{\mathtt{C}}, n_{\mathtt{G}}, n_{\mathtt{T}})$, of the given context. When the length of synthetic sequence reaches the average length of the sequences within the family, the generation procedure is stopped. We generated as many synthetic sequences as the number of families (1,320 synthetic sequences for Rfam, 135 for RDP, and 464 for Greengenes): one synthetic sequence per family (Figure 3.6).

3.2 Results

3.2.1 Dataset preparation and experimental setup

We used function-based and taxonomy-based RNA families. For function-based dataset, Rfam, which consists of functional families such as, rRNAs, tRNAs, miRNAs, siRNAs, and cis-regulatory elements, was used. For taxonomy-based datasets, RDP, Greengenes, and SILVA from different database sources, and Artificial and Diver-

gent from pyrosequencing, were used (Table 3.1). As preprocessing, we excluded sequences containing symbols other than `ACGT(U)` and sequence families with fewer than ten sequences for 10-fold cross validation. Since some methods compared could not handle large data in a reasonable time, we reduced the data as follows: For Greengenes and SILVA, we modified the samples by cd-hit-est [64] using the 97% similarity threshold [95]. For RDP, we downloaded the public datasets preprocessed by mothur. Then, for Rfam and rRNA databases, we limited the maximum number of sequences in each family to 2,000 by random sampling. Sequence-basis and family-basis expanding dataset (Table 3.2) were prepared by randomly selecting sequences and families from large-scale original Greengenes.

We implemented NASCUP in C++ and measured its performance in terms of runtime (*i.e.*, modeling time plus classification time), and classification accuracy with 10-fold cross validation, using a Linux machine (Ubuntu 12.04, 2.2 GHz Intel Xeon E5-4620, and 512 GB memory) without any parallelization. For comparison, we also prepared the implementations of four main alternatives: BLAST, HMMER, RDP, and USEARCH, and five alternatives for additional comparisons: Phymm, gzip, UBLAST, caBLAST, and BLAT. For BLAST and its variants (USEARCH, UBLAST, caBLAST, and BLAT), the class of a query sequence was determined by best-hit. For gzip, the class was determined by the smallest difference of compressed representation between the family data with and without the query sequence.

3.2.2 Classification Performance and Scalability

To see the effectiveness of NASCUP in terms of classification performance and scalability, we have performed a set of comprehensive experiments on real sequence datasets and asymptotically expanding datasets.

The real sequence datasets used in our experiments came from a variety of sources,

Table 3.1: Details of the dataset used in the experiments. We used function-based RNA families named Rfam, taxonomy-based ribosomal RNA families named RDP, Greengenes, and SILVA, and pyrosequencing data named Artificial and Divergent. The datasets had diverse characteristic: AIFD from 0.12 to 0.33, number of families from 23 to 1,320, and sequence length from 20 to almost 5,000. The composition of the families was formed on a completely different purpose by the functional base and the taxonomy base

classification category of the dataset		ID	dataset name (version)	AIFD[‡]	# families[*]	# total[†] sequences	sequence length	ground truth
functional non-coding RNA		RF	Rfam (11.0) [35]	0.33	1,320	170,881	20–1,875	accession
microbial taxonomy	rRNA database (16S, 18S, 23S/28S)	RD	RDP (10.0) [17, 90]	0.08	134	3,838	320–1,833	taxonomy (genus level)
		GG	Greengenes (13.5) [23]	0.12	464	23,142	1,254–2,146	
		SS	SILVA-SSU (119.1) [82]	0.15	313	17,625	902–3,749	
		SL	SILVA-LSU (119) [82]	0.21	107	4,593	1,900–4,954	
	pyrosequencing data (16S rRNA)	AR	Artificial [83]	0.18	60	44,407	40–294	reference sequences
		DV	Divergent [83]	0.14	23	55,466	38–521	

[*] the number of families with more than 10 preprocessed sequences

[†] the total number of sequences after the preprocessing

[‡] average intra-family distance (the normalized pairwise distance between the sequences within a family), ranging from 0 to 1, the AIFD close to zero means a family of similar sequences

Table 3.2: Details of the expanding datasets used in the scalable experiments. Expanding datasets were prepared in two types, sequence-basis and family-basis. Sequence-basis expanding dataset has fixed number of 1,040 families and gradually increasing database sequences. Family-basis expanding dataset has every 104 increasing families from the previous phase, and increasing the total number of training sequences as the number of families increased due to the sequences included in the families

Sequence-basis expanding dataset

Phase	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	P-10
# families	1,040	1,040	1,040	1,040	1,040	1,040	1,040	1,040	1,040	1,040
# tr-seqs*	55,717	111,926	161,634	204,618	279,174	336,311	391,978	434,542	495,348	555,969
# te-seqs†	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000

Family-basis expanding dataset

Phase	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	P-10
# families	104	208	312	416	520	624	728	832	936	1,040
# tr-seqs*	10,842	20,339	30,818	40,739	50,457	62,287	72,602	83,847	93,984	106,003
# te-seqs†	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
# new‡ families	104	104	104	104	104	104	104	104	104	104
# new‡ tr-seqs	10,842	9,497	10,479	9,921	9,718	11,830	10,315	11,245	10,137	12,019

* the number of training sequences as used database

† the number of query sequences for classification test

‡ the number newly added families or sequences from the previous expanding phase

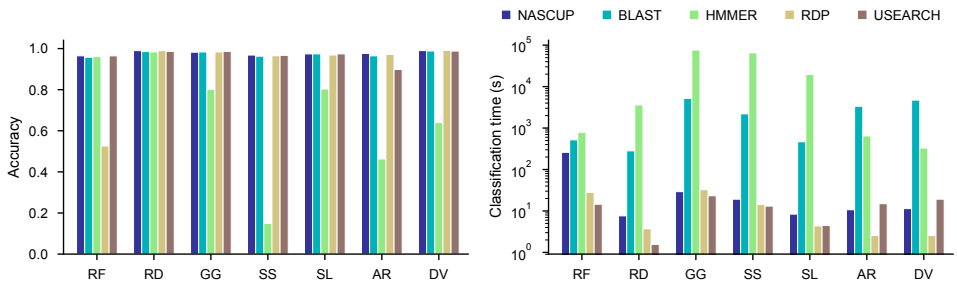

Figure 3.2: Performance comparison in terms of accuracy and classification time. NASCUP and the four main alternatives were experimented on a variety of family type sequence datasets. All the values are the average of 10-fold cross validation. The standard deviation values for accuracy across 10-folds are too small to be represented and the average of standard deviation over the datasets are 0.0027, 0.0029, 0.0276, 0.0031, and 0.0032, respectively, in the order represented in the figure. The classification time represented with log scale. [Each result represented in the order of NASCUP, BLAST, HMMER, RDP, and USEARCH.]

and their families are organized on a functional or taxonomic basis with varying similarity degrees (Table 3.1). Even with this diversity, NASCUP and BLAST showed robust performance over all datasets, maintaining mean accuracy over 0.95, unlike the other methods whose performance varied often significantly over data. HMMER showed good performance on RF but not in microbial datasets, and RDP worked well on microbial datasets but it showed unsatisfactory results on functional RNAs. NASCUP, RDP, and USEARCH ran significantly (often by orders of magnitude) faster for most of the data. The overall performance of RDP and USEARCH, however, did not match that of NASCUP in terms of (both arithmetic and geometric) (Figure 3.2).

Besides the four main alternatives, we compared NASCUP with other alternatives in terms of accuracy and running time (Table 3.3 and 3.4). Regardless of the diversity of dataset, NASCUP showed a high level of robust accuracy and it showed the best performance in term of the average and geometric mean accuracy. In particular, NASCUP showed the highest accuracy on RF dataset, which was the most difficult

classification dataset with the largest number of classes and the widest intra-family sequence distance (AIFD). More than half of the compared methods (UBLAST, BLAT, caBLAST, RDP, gzip, uclust, and mothur) showed unsatisfactory results under 80% of accuracy on RF. NASCUP and BLAST maintained all accuracy over 95% over all datasets considered, unlike the other methods whose performance varied often significantly over data. NASCUP, RDP, and USEARCH ran significantly (often by orders of magnitude) faster for most of the data.

With advances in sequencing and bioinformatics technology, sequencing data is over flooding and database is being updated periodically. The demand for scalability has become more important than ever. As the demand for scalability, we prepared two types of expanding datasets (Table 3.2); one for expanding the number of DB sequences (sequence-basis) and the other for expanding the number of families (family-basis) and huge size of original silva dataset.

In a sequence-basis expansion, the modeling time of NASCUP grew as the number of DB sequences increases. On the other hand, the classification time grew only marginally since it is almost independent of the number of sequences in a family, once the modeling has been completed (Figure 3.3 a-top). The total runtime of NASCUP is orders of magnitude lower than that of BLAST regardless of the data size. For the smallest data, USEARCH was faster than NASCUP, but as the dataset size grew, NASCUP consistently outperformed USEARCH in running time, and the performance gap between the two became wider (Figure 3.3 a-bottom and Table 3.5).

In a family-basis expansion (by increasing every 104 new families to existing families), the classification time of NASCUP grew as the number of families increases. On the other hand, the modeling time did not increase because it performed the modeling procedure only for the newly added families (Figure 3.3 b-top). Likewise in NASCUP , HMMER, and RDP, the generative model-based methods, showed

Table 3.3: Classification accuracy comparison. Each value in the seven datasets is the average of 10-fold cross validation and the last two columns are the average and geometric mean of the averaged values. (pink shade: best in column; grey shade: unsatisfactory (less than 80% accuracy))

method/data	RF	RD	GG	SS	SL	AR	DV	average	geomean
NASCUP	**96.5%**	**99.0%**	98.2%	**96.9%**	97.4%	**97.6%**	99.0%	**97.8%**	**97.8%**
BLAST	95.8%	98.5%	98.3%	96.3%	**97.4%**	96.5%	98.8%	**97.4%**	**97.4%**
USEARCH	96.5%	98.6%	**98.6%**	96.7%	97.4%	89.8%	98.8%	96.6%	96.6%
UBLAST	79.9%	98.5%	97.9%	95.9%	97.1%	96.4%	98.7%	94.9%	94.7%
BLAT	79.1%	97.2%	92.1%	92.3%	95.2%	94.7%	98.9%	92.8%	92.6%
caBLAST	39.4%	97.1%	86.9%	90.5%	93.9%	95.5%	97.0%	85.8%	82.6%
HMMER	96.1%	98.4%	80.1%	14.9%	80.3%	46.3%	64.0%	68.6%	59.7%
RDP	52.6%	99.0%	98.3%	96.5%	96.9%	97.1%	**99.1%**	91.4%	89.5%
Phymm	93.6%	77.5%	76.7%	39.5%	93.2%	95.0%	98.9%	82.1%	79.0%
gzip	62.7%	96.3%	90.3%	80.1%	77.6%	80.9%	96.3%	83.5%	82.7%
QIIME2	83.3%	98.7%	97.1%	94.2%	95.6%	97.0%	99.0%	95.0%	94.8%
UCLUST	23.7%	97.0%	94.4%	85.4%	72.6%	96.8%	98.8%	81.3%	74.6%
Mothur	52.2%	99.0%	98.2%	96.3%	96.8%	95.1%	99.0%	90.9%	89.1%

Table 3.4: Total runtime (classification time) comparison. Each value in the table including average and geomean is the log transformed time result (unit: log_{10} of second). The classification time for BLAST and its variants is the same as the total runtime, since there is no separate modeling process. For HMMER, the preprocessing time for multiple sequence alignment was excluded.

method/data	RF	RD	GG	SS	SL	AR	DV	average	geomean
NASCUP	2.5 (2.4)	1.2 (0.9)	1.7 (1.5)	1.5 (1.3)	1.2 (0.9)	1.6 (1.0)	1.6 (1.1)	1.9 (1.7)	1.6(1.3)
BLAST	2.7	2.5	3.7	3.3	2.7	3.5	3.7	3.4	3.2
USEARCH	1.2	0.2	1.4	1.1	0.6	1.2	1.3	1.1	1.0
UBLAST	2.5	2.8	4.2	3.6	2.6	4.3	4.5	4.0	3.5
BLAT	2.6	3.5	4.8	4.5	3.6	3.3	3.3	4.2	3.7
caBLAST	4.8	3.3	4.5	4.3	3.4	4.4	4.7	4.5	4.2
HMMER	2.9 (2.9)	3.6 (3.6)	4.9 (4.9)	4.8 (4.8)	4.3 (4.3)	2.8 (2.8)	2.5 (2.5)	4.4 (3.7)	3.7(3.7)
RDP	1.6 (1.4)	0.9 (0.6)	1.8 (1.5)	1.6 (1.2)	1.3 (0.6)	0.8 (0.4)	0.8 (0.4)	1.4 (1.1)	1.3(0.9)
Phymm	3.1 (3.0)	1.8 (1.5)	2.8 (2.7)	2.6 (2.5)	2.0 (1.7)	1.7 (1.3)	1.7 (1.0)	2.5 (2.4)	2.2(2.0)
gzip	5.2 (5.2)	3.0 (3.0)	4.3 (4.3)	4.1 (4.1)	3.2 (3.2)	3.9 (3.9)	4.0 (4.0)	4.5 (4.5)	4.0(4.0)
QIIME2	2.3 (2.0)	1.3 (0.8)	1.8 (1.2)	1.7 (1.1)	1.4 (0.9)	1.4 (0.9)	1.4 (0.9)	1.8 (1.4)	1.6 (1.1)
UCLUST	1.7	0.5	1.1	1.3	1.3	1.0	1.1	1.3	1.2
Mothur	2.5	1.2	2.1	2.0	1.6	1.4	1.5	2.0	1.8

a similar pattern (Figure 3.3 b-bottom and Table 3.6). Unlike generative models, however, discriminative learning models, such as SVM, random forest, require full training even though only one new family is discovered.

Classification time for alignment-based methods depends on the total of number sequences in database, regardless of the number of families, since there is no modeling process. On the other hand, classification time for model-based methods depends on the number of families mostly, since they compare a query sequence to model. In case of the modeling time, they only take time to build models for the newly added families because they can build a generative model for each family independently. The runtime difference between sequence-to-sequence and sequence-to-model approaches becomes larger as the database size increases, even though there is no big difference in the early stage.

For scalable test on huge size datasets, we used original SILVA (SSU, LSU) (version 132) [82] without data compaction. SSU and LSU have 1,861 and 587 families at a genus-level, and have 417,387 and 82,087 sequences, respectively. We tested microbiome analysis package QIIME on original SILVA; QIIME 2 Naive Bayes and QIIME 1 taxonomy assignment of UCLUST and Mother are used. As shown in Table 3.7, NASCUP depth 7 showed the highest accuracy in both datasets in a reasonable runtime.

3.2.3 Other classification features and general purposes of NASCUP

To see the feasibility as a preprocessor and robustness in noisy datasets, and to identify the advantage as a generative model that can be used for general purposes, we performed additional experiments.

To overcome the vulnerability to the scalability of sequence-to-sequence approaches, the alignment-based methods may require a model-based classifier as a preprocessor.

Table 3.5: Details of the runtime in the sequence-basis expanding datasets. Since BLAST and USERACH have no modeling procedure, the total time was the same as the classification time. Due to the excessive running time, the classification time for BLAST and HMMER was measured proportionally from the time based on ten number of test sequences

	Phase	# families	# tr-seqs	NASCUP	BLAST	HMMER	RDP	USEARCH
modeling time (s)	P-1	1,040	55,717	36	-	759	59	-
	P-2	1,040	111,926	65	-	781	107	-
	P-3	1,040	161,634	90	-	800	151	-
	P-4	1,040	204,618	111	-	819	189	-
	P-5	1,040	279,174	149	-	794	256	-
	P-6	1,040	336,311	179	-	890	306	-
	P-7	1,040	391,978	206	-	902	362	-
	P-8	1,040	434,542	238	-	909	402	-
	P-9	1,040	495,348	291	-	843	453	-
	P-10	1,040	555,969	325	-	864	518	-
classification time (s)	P-1	1,040	55,717	67	27,794	369,416	86	132
	P-2	1,040	111,926	68	54,182	368,257	81	394
	P-3	1,040	161,634	69	76,587	366,184	78	507
	P-4	1,040	204,618	71	95,389	364,022	78	630
	P-5	1,040	279,174	70	133,472	336,251	77	878
	P-6	1,040	336,311	73	160,358	378,049	77	1,043
	P-7	1,040	391,978	74	183,837	371,104	78	1,166
	P-8	1,040	434,542	79	203,342	367,895	82	1,285
	P-9	1,040	495,348	87	233,456	329,629	81	1,501
	P-10	1,040	555,969	88	256,936	332,026	80	1,691
total time (s)	P-1	1,040	55,717	104	27,794	369,416	144	132
	P-2	1,040	111,926	132	54,182	368,257	187	394
	P-3	1,040	161,634	159	76,587	366,184	229	507
	P-4	1,040	204,618	182	95,389	364,022	268	630
	P-5	1,040	279,174	219	133,472	336,251	333	878
	P-6	1,040	336,311	252	160,358	378,049	383	1,043
	P-7	1,040	391,978	280	183,837	371,104	440	1,166
	P-8	1,040	434,542	317	203,342	367,895	484	1,285
	P-9	1,040	495,348	378	233,456	329,629	535	1,501
	P-10	1,040	555,969	414	256,936	332,026	599	1,691

Table 3.6: Details of the runtime in the family-basis expanding datasets. Since BLAST and USERACH have no modeling procedure, the total time was the same as the classification time. Due to the excessive running time, the classification time for BLAST and HMMER was measured proportionally from the time based on ten number of test sequences. The model-based methods produce a model for each family independently, so the modeling time was measured for 104 new families added at each expanding phase

	Phase	# families	# tr-seqs	NASCUP	BLAST	HMMER	RDP	USEARCH
modeling time (s)	P-1	104	10,842	12	-	75	12	-
	P-2	208	20,339	12	-	72	11	-
	P-3	312	30,818	12	-	75	12	-
	P-4	416	40,739	12	-	75	11	-
	P-5	520	50,457	12	-	75	11	-
	P-6	624	62,287	13	-	76	13	-
	P-7	728	72,602	12	-	75	11	-
	P-8	832	83,847	13	-	75	12	-
	P-9	936	93,984	12	-	75	11	-
	P-10	1,040	106,003	14	-	76	13	-
classification time (s)	P-1	104	10,842	9	1,406	6,718	3	7
	P-2	208	20,339	10	2,220	14,832	7	14
	P-3	312	30,818	12	3,155	24,860	8	22
	P-4	416	40,739	13	4,006	34,355	12	29
	P-5	520	50,457	15	4,922	39,643	15	34
	P-6	624	62,287	16	5,850	46,091	20	43
	P-7	728	72,602	18	6,762	55,132	23	51
	P-8	832	83,847	19	7,795	57,750	22	59
	P-9	936	93,984	20	8,636	74,701	25	65
	P-10	1,040	106,003	21	9,677	76,838	30	87
total time (s)	P-1	104	10,842	21	1,406	6,793	15	7
	P-2	208	20,339	22	2,220	14,904	17	14
	P-3	312	30,818	24	3,155	24,935	19	22
	P-4	416	40,739	25	4,006	34,430	23	29
	P-5	520	50,457	27	4,922	39,718	26	34
	P-6	624	62,287	29	5,850	46,166	33	43
	P-7	728	72,602	30	6,762	55,207	34	51
	P-8	832	83,847	32	7,795	57,824	34	59
	P-9	936	93,984	32	8,636	74,776	36	65
	P-10	1,040	106,003	35	9,677	76,914	43	87

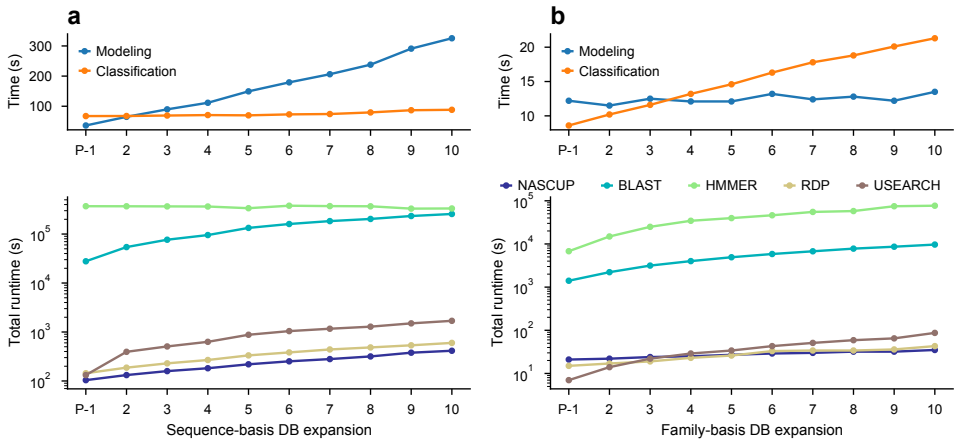

Figure 3.3: Scalability experiments on sequence-basis and family-basis expanding dataset. The top two plots represent modeling vs classification time of NASCUP. The bottom plots represents the total runtime (unit:log) comparison of NASCUP and the four main alternatives. The column represents scalable results **(a)** on sequence-basis expansion datasets and **(b)** on family-basis expansion datasets.

Table 3.7: Experimental results in terms of accuracy and time on original SILVA (SS, SL) datasets. NASCUP-7 is comparable to RDP whose default k is 8, NASCUP-6 is the default parameter of NASCUP. Microbiome analysis package QIIME is used for taxonomy genus-level classification; QIIME 1 taxonomy assignment of UCLUST and Mothur, and QIIME 2 (2018-04) Naive Bayes are used.

method/data	accuracy		time(s)	
	O-SSU	O-LSU	O-SSU	O-LSU
NASCUP-7*	**99.04%**	**98.68%**	1343.80	180.37
NASCUP-6*	98.79%	98.23%	1139.39	145.23
RDP	98.76%	98.28%	2612.79	566.69
QIIME2	97.50%	96.73%	2119.97	370.37
UCLUST	98.05%	95.47%	525.62	109.92
Mothur	98.68%	98.23%	9792.79	2275.26

* NASCUP -7 means NASCUP depth of 7, and NASCUP-6 means NASCUP depth of 6

Table 3.8: Accuracy comparison from candidate extraction. Two sections represent different extracting methods about extracting candidate families as varying (1) number of candidates, fixed to k and (2) threshold T on the sum of the posterior probabilities, respectively. Each value in the seven datasets is the average recall of 10-fold cross-validation. In the second section, the numbers in parentheses mean the average number of candidate families

(1) fixed k

k	RF	RD	GG	SS	SL	AR	DV
1	0.9648	0.9901	0.9822	0.9685	0.9741	0.9763	0.9902
3	0.9872	0.9987	0.9986	0.9927	0.9885	0.9867	0.9915
5	0.9898	0.9990	0.9993	0.9961	0.9915	0.9882	0.9927
10	0.9926	0.9990	0.9999	0.9977	0.9948	0.9900	0.9942

(2) threshold T on sum of the posterior probabilities

T	RF	RD	GG	SS	SL	AR	DV
0.95	0.9725 (1.097)	0.9904 (1.001)	0.9828 (1.001)	0.9691 (1.002)	0.9743 (1.001)	0.9823 (1.014)	0.9904 (1.002)
0.99	0.9759 (1.264)	0.9904 (1.001)	0.9832 (1.002)	0.9696 (1.004)	0.9743 (1.001)	0.9840 (1.020)	0.9905 (1.004)
0.999	0.9798 (1.819)	0.9906 (1.002)	0.9838 (1.004)	0.9702 (1.006)	0.9743 (1.002)	0.9854 (1.030)	0.9907 (1.006)
0.9999	0.9830 (3.004)	0.9906 (1.002)	0.9844 (1.006)	0.9706 (1.008)	0.9747 (1.003)	0.9859 (1.053)	0.9908 (1.008)

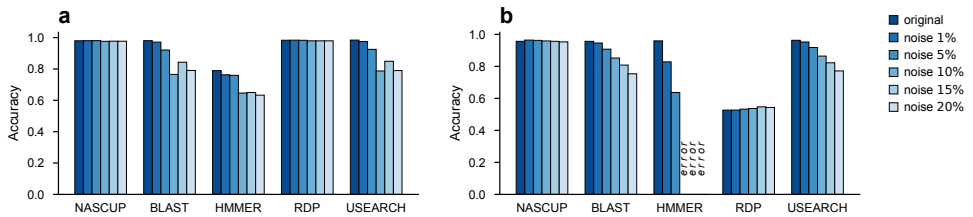

Figure 3.4: Performance changes as mislabeling noise ratio increases. Classification performance on the mislabeling noise mixed **(a)** GG and **(b)** RF datasets. The mislabeling noise ratio increased from 1% to 20% and the bar color gradually faded as the noise ratios increased. If sequences in a class are too diverse, HMMER may not perform the multiple sequence alignment (MSA) properly due to no consensus columns between sequences. HMMER was not able to build models on RF datasets of higher noise ratios with this problem.

To see the feasibility as a preprocessor, we checked the accuracy of NASCUP as extending the top rank. Extending the top rank to 10, NASCUP could acquire over 99% accuracy in all datasets used (Table 3.8). If we use top 10 families as candidates we could reduce the search space enormously.

Sequence label is assigned by domain experts manually or bioinformatic programs, so existing database may have sequences that are labeled incorrectly. We tested the classification robustness against databases with mis-labeled sequences, by increasing the mis-labeling ratio from 1% to 20%. Each sequence is assigned a probability for a label change, and if the probability is higher than the noise ratio, the label of the sequence is changed to an arbitrary label. NASCUP showed strong robustness with little performance degradation even in 20% of noise database, unlike other methods whose performance degraded as the noise ratio increases (Figure 3.4).

For discriminative classification approaches, one-class classification is a very challenging problem because it should identify whether or not a sequence belongs to the target class, with only the target class data given. NASCUP builds a generative model for each family independently; it does not need any adversarial discriminative data. we tried one-class classification for outlier detection in Staphylococcus and mi-

croRNAs and compared its performance with HMMER. NASCUP showed explicitly distinct cross-entropy values for outliers and PR curves of higher AUC compared to HMMER (Figure 3.5).

To see the performance on generative aspect, we generated four types of synthetic sequences according to the combination of models (NASCUP and k-mer) and sampling methods (universal probability (KT) and maximum likelihood (ML)) as increasing k from 2 to 10. The evaluation of the generated sequences was judged by classification accuracy indirectly; We evaluated the quality of generated sequences with the five main classifiers (BLAST, HMMER, RDP, USEARCH, and NASCUP). As the k-size increases, while most of the k-mer based sequences suffered from performance degradation, NASCUP based sequences showed stable performance without such a big degradation. The KT sampling showed better classification performance than the ML sampling in most of the experiments. The classification performance of the NASCUP model-KT sampling combination increased resulting in a comparable accuracy from the real datasets. The quality of synthetic sequences was affected by both of the modeling and sampling methods (Figure 3.6). Through these experiments, we reaffirmed the superiority of the NASCUP model and universal probability.

3.3 Discussion

The proposed NASCUP method performs model-based classification using the universal probability assignment for context-tree models from information theory.

NASCUP can be summarized as follows:

- NASCUP is a model-based classifier that builds a context-tree model for a family of sequences and predicts a family of a query sequence by using universal probability.

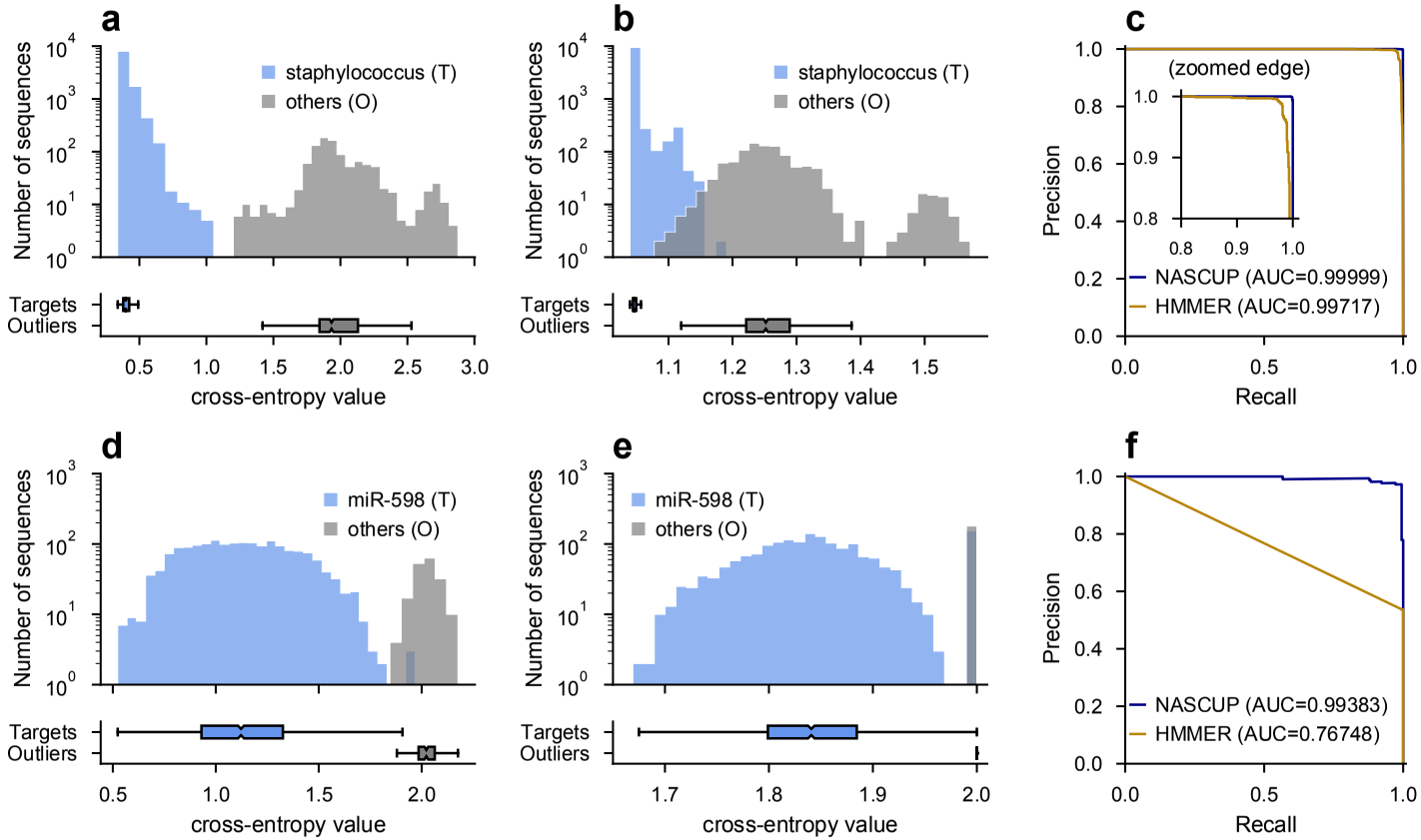

Figure 3.5: Outlier detection with cross-entropy value. Distribution of predicted cross-entropy values with histogram and boxplot of **(a, d)** NASCUP and **(b, e)** HMMER on dataset that mixed a 10:1 ratio of targets and outliers. **(c, f)** Precision-recall (PR) curves and their area under the curve (AUC) values of NASCUP and HMMER. The top three plots are experimental results on Staphylococcus mixed with outliers from GG dataset, and the bottom three plots are results on miR-598 mixed with outliers from RF dataset.

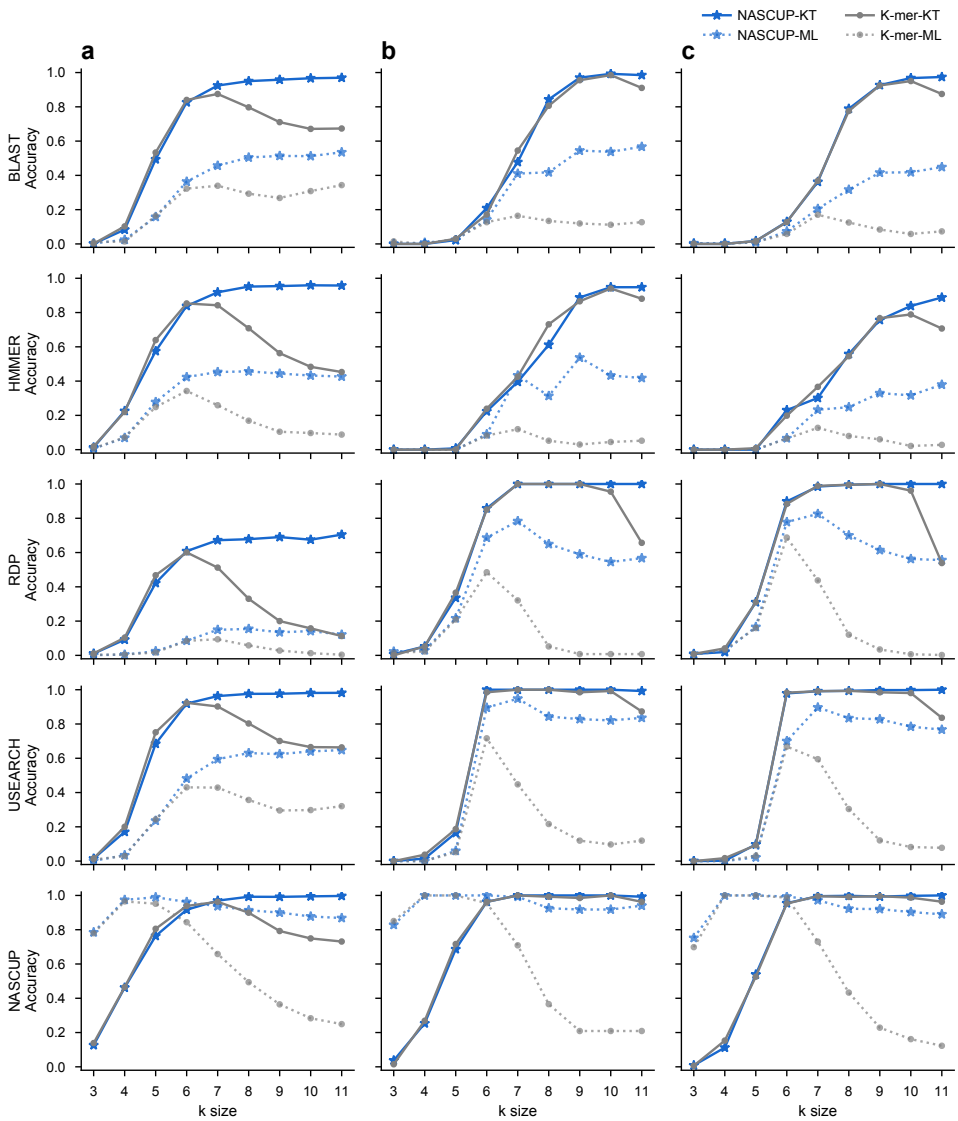

Figure 3.6: Synthetic sequence generation and its quality assessment with classifiers. We generated four types of synthetic sequences by a combination of the modeling methods (NASCUP and k-mer) and sampling methods (universal probability (KT) and maximum likelihood (ML)). With regard to the modeling method, it is represented in different color; NASCUP model as blue color and k-mer model as gray color. With regard to the sampling method, it is represented in line style; KT sampling as solid line and ML sampling as dotted line. The column represents the classification results according to the tested dataset of **(a)** RF, **(b)** RD, and **(c)** GG, and the row represents the classification results according to the methods in order of BLAST, HMMER, RDP, USEARCH, and NASCUP. The k size changes from 3 to 11 (meaning depth, d, from 2 to 10).

- NASCUP delivers robust accuracy comparable to the sequence-to-sequence approaches on diverse datasets, and NASCUP operates in a scalable style as the sequence-to-model approach suitable for large-scale expanding datasets. It produces a generative model for each family independently (suitable for family-basis incremental learning or continual learning), so it does not need a full training required for a discriminative model.

- NASCUP can be a preprocessor to reduce search space enormously for sequence-to-sequence approaches by top candidate class selection.

- NASCUP produces a generalized model which is robust even in a noisy dataset and can be used for a variety of purposes, such as outlier detector and synthetic sequence generator with the advantages of the generative model.

As a model-based method, NASCUP can be easily adapted to test similarity between families or a membership of a sequence or a sequence family in a group of families. We expect that this modified version of NASCUP can be applied to problems with inherent hierarchical structures such as taxonomy classification and phylogenetic tree construction. A phylogenetic tree is typically constructed by measuring sequence similarities using heuristics such as progressive alignment and then by applying linkage methods to produce a dendrogram structure. In this conventional approach, results may vary (often dramatically) according to which linkage method is used. In contrast, the modified version of NASCUP would allow direct model-to-model comparison, alleviating the aforementioned limitations of alignment–linkage methods. Going one step further, we expect that the modified NASCUP, fully utilizing the simple Bayesian update structure of universal probability assignments in the classification stage, will be applicable to incremental clustering, in which an initial model is constructed from close sequences and then is incrementally augmented by

adding additional sequences.

Instead of unidirectional contexts (that precede symbols), we can use bidirectional contexts (that precede and succeed symbols). More generally, the directionality of nucleotide sequences or the lack thereof can be incorporated into modeling and classification stages. Additionally, the maximum depth of context-tree models can be adjusted more flexibly in a data-dependent manner. A similar technique has been developed in the context of data compression [114], which is expected to be applicable in our problem. The use of bidirectional contexts and adaptive maximal depths is expected to boost accuracy further at some cost of time efficiency.

Chapter 4

Siamese CNN Based Interaction Learning

Interaction represents an action that occurs between two entities having similar functions or sharing of metabolic pathways [8, 74, 92]. There are various interactions such as protein-protein [30, 43, 88], compound-protein [103], miRNA-mRNA [59], and chemical-chemical interactions (CCI) [55].

Many studies based on CCI information have been conducted and include the following: Studies predicting compound synergism, which tries to search the therapeutic efficacy of molecular combination based on interacting chemicals [113]; predicting chemical toxicity or side effect on the assumption that interacting chemicals will share similar toxicities [12, 13]; discovering new candidate drugs by graphically linking interacting chemicals that are most similar to commercialized drugs [14]; identifying the biological functions of chemicals on the assumption that interacting chemicals participate in the same metabolic pathways [40].

As listed above, CCI information has been used for many purposes. However, CCI has been built from text mining, biological experiments, similarities between

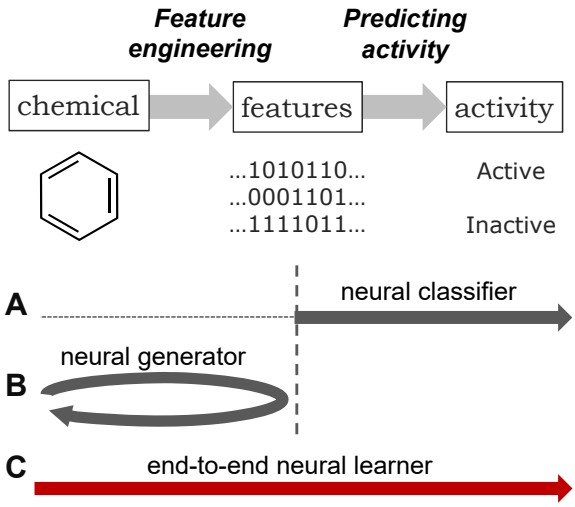

Figure 4.1: Two-stage chemical analysis: feature engineering and predicting activity. Three types of neural network based learning approaches are shown: (A) neural classifier, whereby neural network based learning is applied to the back-end, (B) neural generator, whereby neural network based learning is applied to the front-end, and (C) end-to-end neural learner, whereby neural network learning is applied from the front-end to back-end (our approach).

chemicals, and other database information, and has not been predicted on a learning basis. In this paper, we propose a learning-based CCI prediction method.

Typical learning-based chemical (drug) analyses consist of two stages, *i.e.*, feature engineering and predicting activity, as shown in Figure 4.1. In drug analyses, the feature is also called the descriptor, fingerprint, or molecular representation.

The feature engineering stage extracts meaningful and useful features from chemicals. The vast amount of information, which has been manually designed by domain experts, varies from simple counts (*e.g.*, the number of atoms, rings, and bonds) to topological properties (*e.g.*, atom pairs, shapes, and connectivities) [104]. Extracted features have been used for subsequent analysis and have had a significant impact on predictive performance. Various studies have been proposed to find the essential properties of chemicals, and more than 5,000 versatile features have been identified

[104] (*e.g.*, PubChem, ECFP, and MACCS fingerprint).

The predicting activity stage uncovers the relationship between extracted features and biological activities such as absorption, distribution, metabolism, excretion, and toxicity (ADMET) [100, 106]. Machine learning methods such as support vector machine (SVM) [19, 24] and random forest (RF) [98] have been successfully applied to study the relationship and to handle large amounts of data.

In recent years, the neural network based learning methods have advanced rapidly and have achieved great results for challenging problems, such as natural language processing [6, 65, 97], speech and image recognition [22, 34, 38, 53, 118], and computational biology [29, 58, 60, 62, 79]. The achievements of neural network technologies in the drug field [20, 69, 84, 103] have also attracted people's attention. Neural network based learning approaches in drug analyses can be categorized into three cases(see Figure 4.1):

(A) **Neural classifier:** Neural network based learning is utilized as a back-end classifier, learning from manually crafted chemical features. The neural network based classifier has attracted considerable interest through the Kaggle competition of the "Merck Molecular Activity Challenge" and its subsequent quantitative structural activity (QSAR) predictions [20, 84]. The Kaggle competition and its subsequent predictions have greatly improved prediction performance by using multi-tasking techniques based on neural network learning from provided features. In addition, prediction of chemical-protein interaction [103] (for drug discovery, network pharmacology, and drug/protein target identification) showed boosted results based on neural network architecture from fingerprints. The proposed neural network based classifiers have improved prediction performance based on the well-crafted chemical features.

(B) **Neural generator:** Neural network based learning is utilized as a front-end generator for drug generation. The drug generation extracts the hidden features from the chemical and regenerates the chemicals again from the extracted features. Generative neural network models, such as a variational autoencoder [49] and a recurrent neural network [39], have been utilized for drug generation [33, 91]. Both models use the simplified molecular input line entry system (SMILES) [100] as input to extract the hidden features and regenerate the SMILES string. Neural network based learning is used as a neural generator between chemical strings and extracted features.

(C) **End-to-end neural learner:** Neural network based learning is utilized as an end-to-end learner that includes the entire process from feature engineering to predicting biological activity. It should extract the hidden features from the original chemical input and exploit the extracted features to predict the biological activity. Owing to the difficulties of handling original chemical information, the end-to-end learning framework was not actively exploited.

In drug (chemical) analyses, neural network based learning is mainly used as described in case (A), *i.e.*, as a neural classifier. However, deep learning is not just for classification, but also for extracting hidden features [70]. The hidden features extracted by neural networks may contain important features unknown to domain experts. With the key features, we can improve predictive performance. For this reason, we propose an end-to-end neural network learning framework, named DeepCCI, for CCI prediction.

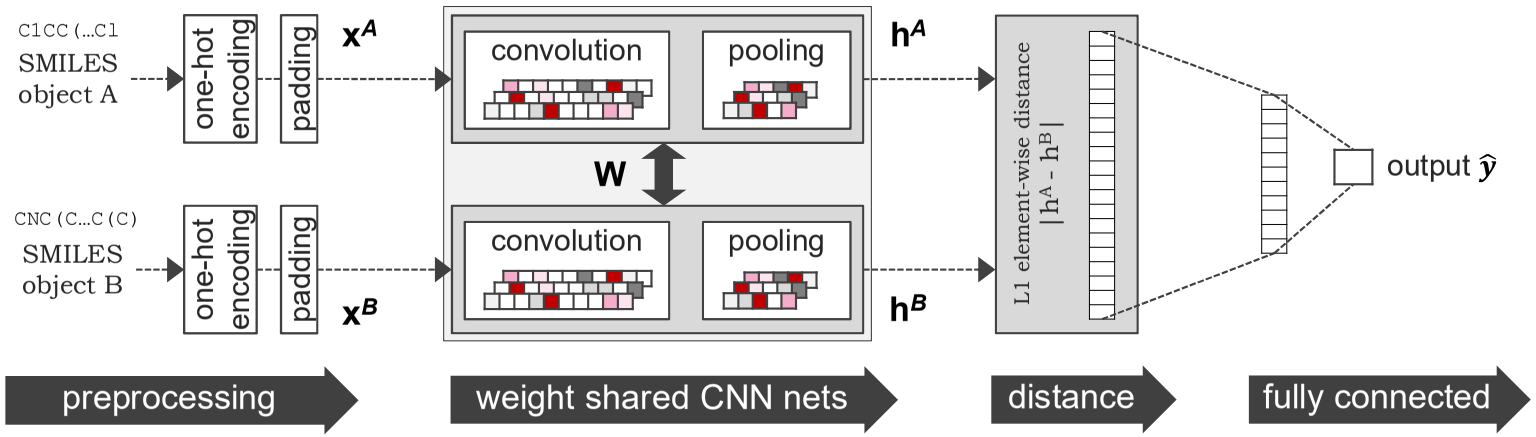

Figure 4.2: Overview of the proposed end-to-end learning methodology. Our method extracts hidden representations from SMILES learning and predicts the interaction through the extracted representations, based on the siamese architecture. Inputs are one-hot encoded and zero padded (or truncated) to the maximum length. The preprocessed inputs are fed into shared 1D-CNNs for transforming into hidden representations. L1 element-wise distance between the hidden representations for objects A and B are measured and then fed into fully-connected layers to predict the interaction. The output is the predicted probability of the interaction between objects A and B.

4.1 Methods

Figure 4.2 shows an overview of the proposed DeepCCI. Our method presents end-to-end SMILES learning for CCI. It takes SMILES strings as inputs $\mathbf{x}^A$ and $\mathbf{x}^B$ for objects A and B, and then produces an interaction probability $\hat{y}$. The structure is divided into three stages. The first stage is for preprocessing SMILES inputs, the second stage is for learning latent hidden representations through weight shared 1D-CNNs, and the third stage is for distance and interaction prediction through L1 element-wise distance measure and fully connected layers. Algorithm 1 shows the overall procedure of DeepCCI.

4.1.1 Notations

The notations used in our paper are outlined below.

- $\mathcal{I}(A, B)$: interaction between objects A and B
- $\mathcal{X}$: SMILES character set, $\mathcal{X} = \{\texttt{C}, =, \texttt{(}, \texttt{)}, \texttt{O}, \texttt{N}, \texttt{1}, \texttt{2}, \cdots\}$
- $\mathbf{x}$: SMILES string, $\mathbf{x}^A$ and $\mathbf{x}^B$ for objects A and B
- x_i: i-th character of $\mathbf{x}$, $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, $x_i \in \mathcal{X}$
- $\mathbf{h}$: hidden representations, $\mathbf{h}^A$ and $\mathbf{h}^B$ for objects A and B
- h_i: i-th element of $\mathbf{h}$, $\mathbf{h} = (h_1, h_2, \cdots, h_n)$
- λ: maximum length of SMILES
- $\mathcal{F}$: number of filters
- k: length of filter (kernel size)
- $\mathbf{W}$: learning weights

Algorithm 1 Pseudo-code of proposed DeepCCI

Input: $\mathbf{x}^A = \left(x_1^A, x_2^A, \cdots, x_\lambda^A\right)$ and $\mathbf{x}^B = \left(x_1^B, x_2^B, \cdots, x_\lambda^B\right)$ ▷ $\mathbf{x}^A$, $\mathbf{x}^B$: the encoded and padded/truncated inputs for objects A and B
Output: $\hat{\mathbf{y}}$ ▷ $\hat{\mathbf{y}}$: predicted probability of interaction
repeat
  $\mathbf{h} = \mathbf{Pool}(\sigma_r(\mathbf{Conv}(\mathbf{x})))$ ▷ shared CNNs described in Section 4.1.3
  $\hat{y} = \sigma_s(\mathbf{Fcl}(\sigma_r(\mathbf{Fcl}(|\mathbf{h}^A - \mathbf{h}^B|))))$ ▷ interaction prediction described in Section 4.1.4
    ▷ $\mathbf{h}^A$, $\mathbf{h}^B$: the outputs of shared CNNs from inputs $\mathbf{x}^A$,$\mathbf{x}^B$
  $\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$ ▷ learning objective to minimize the loss $\mathcal{L}(y, \hat{y})$
  $\mathbf{W} = \mathbf{W} - \Delta\mathbf{W}$ ▷ weights update $\Delta\mathbf{W}$ based on averaged $\mathcal{L}$ from mini-batch
until number of epoch reaches n_{epoch}

- $\hat{y}$: predicted probability of interaction
- σ: activation function, σ_s for the sigmoid, σ_r for the rectified linear unit
- N: sample size, N_t for total sample, N_b for mini-batch size

4.1.2 Input Preprocessing

We used input $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ which is represented by 65-character $x_i \in \mathcal{X}$, where $\mathcal{X} = \{\texttt{C}, \texttt{=}, \texttt{(}, \texttt{)}, \texttt{O}, \texttt{N}, \texttt{1}, \texttt{2}, \cdots\}$, $|\mathcal{X}| = 65$ in the SMILES format. Input $\mathbf{x}$ should be converted into a numeric expression for learning.

For the numerical expression, we employ a widely used one-hot encoding scheme, which sets the corresponding single character to '1' and all the others to '0', thereby converting each character x_i into an $|\mathcal{X}|$-dimensional binary vector. An example of

cyclohexane (`C1CCCCC1`) encoded $\mathbf{x}$ is as follows:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} \texttt{C} \\ \texttt{1} \\ \texttt{C} \\ \texttt{C} \\ \texttt{C} \\ \texttt{C} \\ \texttt{C} \\ \texttt{1} \end{bmatrix} = \begin{matrix} \{\texttt{C}, =, \texttt{)}, \texttt{(}, \texttt{O}, \texttt{N}, \texttt{1}, \texttt{2} \cdots\cdots\cdots \} \\ \begin{bmatrix} [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [0,0,0,0,0,0,1,0,\cdots,0,0,0] \\ [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [1,0,0,0,0,0,0,0,\cdots,0,0,0] \\ [0,0,0,0,0,0,1,0,\cdots,0,0,0] \end{bmatrix} \end{matrix}$$

In general, SMILES strings have variable lengths depending on the complexity of the chemical structure. Chemical compounds of a complex structure have relatively longer SMILES strings compared to compounds of a simple structure. To effectively learn the hidden representations for SMILES, we limit the inputs to a certain maximum length, λ. If a sequence is shorter than the maximum length, zero values are pre-padded; otherwise, the sequence is truncated to the maximum length. After the padded or truncated processes, all input lengths are fixed to the maximum length, λ. According to the length distribution, we limits the SMILES strings to a maximum length, λ, as 100 (see Figure 4.3).

According to the encoding scheme and maximum length, input $\mathbf{x}$ is encoded into a $\lambda \times |\mathcal{X}|$-dimensional binary matrix. The input collection is represented by a $N_t \times \lambda \times |\mathcal{X}|$-dimensional binary tensor.

4.1.3 Weight Shared CNN Networks

To learn hidden representations $\mathbf{h}$ from the preprocessed input $\mathbf{x}$ of SMILES strings, we used CNNs. The CNN architecture, used in our method, consists of a convolution layer and a pooling layer.

$$\mathbf{h} = \mathbf{Pool}(\sigma_r(\mathbf{Conv}(\mathbf{x}))) \tag{4.1}$$

The convolution layer plays a 'motif detector' role across the sequence. The preprocessed input $\mathbf{x}$ is convolved with a set of learnable feature maps called filters (or kernels). Different filters may detect motifs of different chemical properties in the SMILES string. Scanning the local region across the input string with a parameter-shared filter may enable recognition of the shift-invariant local pattern.

The output shape of the convolved input $\mathbf{x}$ becomes an $\mathcal{F}\times(\lambda-k+1)$-dimensional matrix, where $\mathcal{F}$ represents the number of filters, and k represents the filter length.

$$\begin{aligned} \mathbf{Conv}(\mathbf{x}) &= (c_{f,i}) \in \mathbb{R}^{\mathcal{F}\times(\lambda-k+1)} \\ c_{f,i} &= \sum_{j=1}^{k}\sum_{s=1}^{|\mathcal{X}|} W_{f,j,s}x_{i+j,s} \end{aligned} \tag{4.2}$$

where f and i are $1 \leq f \leq \mathcal{F}$ and $1 \leq i \leq (\lambda - k + 1)$, respectively. The parameters $\mathcal{F}$ and k of the CNN filter are decided through experiments (see Figure 4.5D and 4.5E).

After the convolution layer, each convolved element is processed with the rectified linear unit (ReLU) [72]

$$\sigma_r(t) = \max(0, t) \tag{4.3}$$

which is an activation function that is not saturated for positive and clamped to zero for negative.

The pooling layer partitions the rectified responses with non-overlapping sub-regions and then outputs the maximum if it is max-pooling. It summarizes within the sub-regions and reduces the spatial size. The $\mathcal{F}$ dimension remains unchanged. A max-pooling length of 6 showed the best performance in our experiments (see Figure 4.5C)

To prevent overfitting and to produce generalization effects, we apply dropout [94] for the last CNN units.

For a symmetric problem, such as a tweets comparison, we exploit the siamese network architecture which reuses the weights between two networks [15, 50]. Likewise, we use shared CNNs to extract the hidden representations h^A and h^B for chemicals A and B from inputs $\mathbf{x}^A$ and $\mathbf{x}^B$.

4.1.4 Distance and Interaction Prediction

Commutative Property

In mathematics, a commutative property means that changing the order does not affect the result (for example, $2 + 5 = 5 + 2$). As in indirect symmetric problems, such as distance or similarity between objects, predicting the interaction between objects A and B is independent of the ordering, namely $\mathcal{I}(A, B) = \mathcal{I}(B, A)$.

However, if the values $\mathbf{x}^A$ and $\mathbf{x}^B$ for objects A and B are simply concatenated,

and then used for learning, the result can be different according to the input order

$$\begin{aligned}
&\mathcal{I}(\mathbf{x}^A, \mathbf{x}^B) \\
&= w_1 x_1^A + \cdots + w_n x^A + w_{n+1} x_1^B + \cdots + w_{2n} x_n^B \\
&\neq w_1 x_1^B + \cdots + w_n x^B + w_{n+1} x_1^A + \cdots + w_{2n} x_n^A \\
&= \mathcal{I}(\mathbf{x}^B, \mathbf{x}^A).
\end{aligned} \tag{4.4}$$

To guarantee the commutative property, we exploit the Siamese [50] architecture suitable for symmetric problems.

Proposed Structure for Commutative Property

Unlike heterogeneous interactions, such as protein-chemical and miRNA-mRNA interaction, CCI is a homogeneous interaction between two chemicals. The probability of homogeneous interaction between objects A and B should be the same regardless of their input order, $\mathcal{I}(A, B) = \mathcal{I}(B, A)$, thereby guaranteeing the commutative property described in Section 4.1.4. However, simple concatenating of the values of objects A and B cannot guarantee the property according to Equation 4.4.

To equally recognize $\mathcal{I}(A, B)$ and $\mathcal{I}(B, A)$, we applied the L1 element-wise distance between hidden representations $\mathbf{h}^A$ and $\mathbf{h}^B$ for objects A and B with different weights

$$\begin{aligned}
\mathcal{I}(A, B) &\approx \mathbf{W}|\mathbf{h}^A - \mathbf{h}^B| \\
&= w_1|h_1^A - h_1^B| + \cdots + w_n|h_n^A - h_n^B| \\
&= w_1|h_1^B - h_1^A| + \cdots + w_n|h_n^B - h_n^A| \\
&= \mathbf{W}|\mathbf{h}^A - \mathbf{h}^B| \approx \mathcal{I}(B, A).
\end{aligned} \tag{4.5}$$

The element-wise distance vector feds into the two fully-connected layers (Fcl).

$$\hat{y} = \sigma_s(\mathbf{Fcl}(\sigma_r(\mathbf{Fcl}(|\mathbf{h}^A - \mathbf{h}^B|)))) \tag{4.6}$$

where $\hat{y}$ is the predicted probability , σ_r is the ReLU in Equation (4.3), and σ_s is the sigmoid activation function

$$\sigma_s(t) = 1/(1 + \exp(-t)) \tag{4.7}$$

which make the output probability ranging from zero to one.

The learning processes proceed to minimize the loss between the true label and the prediction $\mathcal{L}(y, \hat{y})$, where y is the true label and $\hat{y}$ is the predicted probability from Equation (4.6). For the loss function as the learning objective, we use the binary cross entropy

$$\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}). \tag{4.8}$$

Weights are updated in mini-batch units; the loss for the batch unit is averaged by the mini-batch size.

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N_b} \sum_{i=1}^{N_b} \mathcal{L}(y_i, \hat{y}_i) \tag{4.9}$$

where N_b is the mini-batch size, and i represents the sample index.

In the fully-connected layers, we use 128 hidden nodes, batch normalization [42], and a dropout [94] rate of 0.25 for regularization. In our experiments, the entire learning was conducted with the Adam optimizer [48] (learning rate: 0.001, epoch: 100, and mini-batch size: 256). Most of these parameters and network structure were experimentally determined. An epoch of 100 showed stable convergence. We did not use early stopping.

Table 4.1: Details of the datasets used in the experiments. The negative samples were randomly generated from negative-0 to balance them with the number of positives

dataset	CCI900	CCI800	CCI700
# of pos	6,515	53,274	116,020
# of neg	6,515	53,274	116,020
# of total	13,030	106,548	232,040

4.2 Results

4.2.1 Experimental Setup

Dataset

In our experiment, we downloaded the CCI data (version 5.0) from the STITCH [55] database. Each interaction pair in the downloaded CCI data had a confidence score ranging from 0 to 999, which was derived from text-mining, experiments, chemical similarity, and other database information. The higher the confidence score, the more likely the two chemical compounds were to interact; a score of 999 meant strong interaction, and a score of zero meant no interaction between two chemical compounds.

According to the confidence score, three datasets were prepared: CCI900, CCI800, and CCI700 as listed in Table 4.1. Each dataset consists of the same number of positive and negative pairs. The positive pairs of CCI900, CCI800, and CCI700 were selected with a score greater than 900, 800, and 700, respectively. The negative pairs of each dataset were randomly selected with a score of zero to balance them with the number of positive pairs.

Prior to the data preparation, compounds with RDKit [56] errors or SMILES string lengths shorter than 20 or longer than 100 characters were excluded. As shown in Figure 4.3, the lengths of SMILES strings varied from 1 to over 200. Their median length is 35, and the average length is 42.2. Of the SMILES strings, 91%, 95%, and

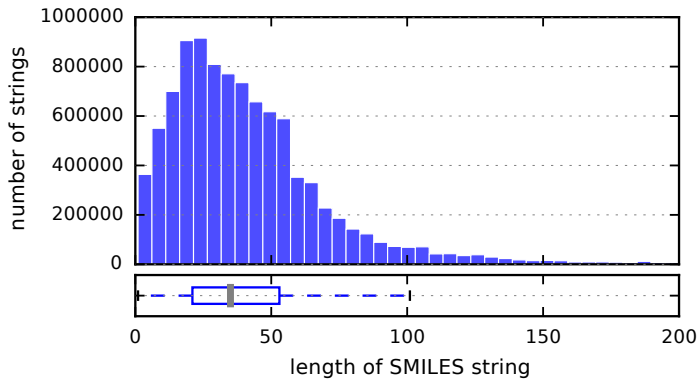

Figure 4.3: SMILES string length distribution: the median length is 35 and the average length is 42.2. Ninety-five percent of the SMILES strings are less than 100 characters in length.

97% of them are less than 80, 100, and 120 characters in length, respectively.

For a fair comparison, we generated an asymmetric dataset, as methods of guaranteeing the commutative property would be advantageous to the symmetric dataset. In the asymmetric dataset, only one of the A-B pair and B-A pair was included; its order switched pair was not included in the dataset.

Input Representation of Chemical Compound

The input representations of chemical compounds were prepared as four kinds: a string type of SMILES and a numerical type of PubChem [110], ECFP [71], and MACCS [25] fingerprints.

The SMILES string was used as input for the proposed end-to-end learning method, DeepCCI. The PubChem, ECFP, and MACCS fingerprints were used as input for the conventional machine learning methods (SVM, RandomForest, and AdaBoost) and a plain neural network classifier (FFNN, see Figure 4.1A).

The PubChem fingerprint represents chemical properties as an 881-dimensional binary vector that has feature information, such as the presence or count of atoms, rings, and nearest neighbor patterns. The ECFP as a kind of circular fingerprint pro-

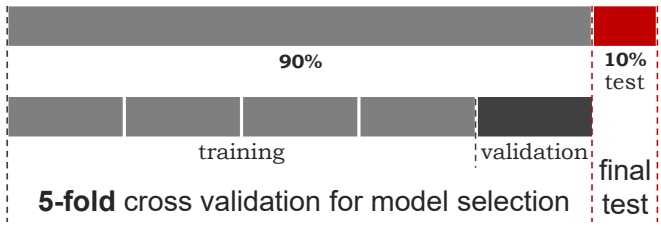

Figure 4.4: Dataset breakdown. The full dataset was randomly divided into the model selection set (90%) and final test set (10%). the model selection set was again partitioned into five sets, for 5-fold cross validation. Through the iterative validation, the model with hyperparameters, showing the best average performance, was selected.

duces circular atom neighborhood features. We used the 2048-dimension features of the ECFP fingerprint. The MACCS consists of 166-dimensional values based on SMARTS. It is relatively small in size but covers most of the major features.

From the chemical IDs listed in the STITCH database, we retrieved the SMILES string and PubChem fingerprint by using PubChemPy [99], and we retrieved ECFP and MACCS fingerprints by using RDKit [56].

An example of PubChem cyclohexane representative is as follows:

- PubChem chemical ID: 8078
- SMILES string: `C1CCCCC1`
- PubChem fingerprint: $110000000110000000000000000\cdots$

  (the 9-th bit is 1 if there are more than two carbon atoms and is 0 otherwise;

  the 10-th bit is 1 if there are more than four carbon atoms and is 0 otherwise)

The fingerprints of two objects were concatenated as a $(2\times|\text{fingerprint dimension}|)$-numerical vector and then used as the inputs for the learning algorithms.

Experimental Configuration

Our dataset consisted of training, validation, and test samples, as represented in the general learning experiments [5]. Figure 4.4 details the configuration of our dataset. The full dataset was divided into two parts, one for model selection and the other for the independent test.

The goal of model selection was to find the optimal hyperparameters. Models with various sets of hyperparameters were trained and then iteratively validated through k-fold cross validation. Optimal hyperparameters, which showed the best average performance in terms of AUC, were selected. In our experiments, we used the 5-fold cross validation.

The remaining test set was used to assess the generalization of the selected model, and compare its performance with other learning methods. The fairness of the independent test could be guaranteed, as the parameters were determined through training and validation independently of the test set.

Our experiments were run on Ubuntu 14.04 (3.5GHz Intel i7-5930K and GTX Titan X Maxwell (12GB)). For the implementation of neural network architecture, we used the Keras library package (version 1.1.1) [15]. For the test of conventional machine learning methods, we used the Scikit-learn library package (version 0.18) [81]

Evaluation Metrics

For the performance comparison, we used eight different evaluation metrics:

- AUC: the area under the ROC curve, which is widely used in case of binary classification to show the overall performance of a binary classifier[1]

[1]The area under the ROC curve which plots $TPR(T)$ versus $FPR(T)$ with varying threshold T, where $FPR = 1 - TPR$

- ACC: accuracy, which is intuitively used as a metric to assess performance[2]
- MCC: Matthews correlation coefficient[3]
- TPR: true positive rate, sensitivity or recall[4]
- TNR: true negative rate or specificity[5]
- PPV: positive predictive value or precision[6]
- NPV: negative predictive value[7]
- F1: F1 score[8]

4.2.2 Effects of Hyperparameter Variation

Five different structure and hyperparameter variation experiments were performed to find the empirical optimal structure and parameters.

First, we conducted two different structural experiments related to the handling of extracted hidden representations. One is focused on the merging structure and the other is focusing on the element-wise distance structure between the hidden representations extracted from object A and B. (The merging structure uses an addition operation to give the same weights to the two hidden feature values.) As shown in Figure 4.5A, the L1 element-wise distance structure showed a better performance than the merging structure.

[2] $ACC = (TP + TN)/(TP + TN + FP + FN)$, where
TP: true positive, FP: false positive, FN: false negative, TN: true negative
[3] $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
[4] $TPR = TP/(TP + FN)$
[5] $TNR = TN/(TN + FP)$
[6] $PPV = TP/(TP + FP)$
[7] $NPV = TN/(TN + FN)$
[8] $F1 = 2TP/(2TP + FP + FN)$

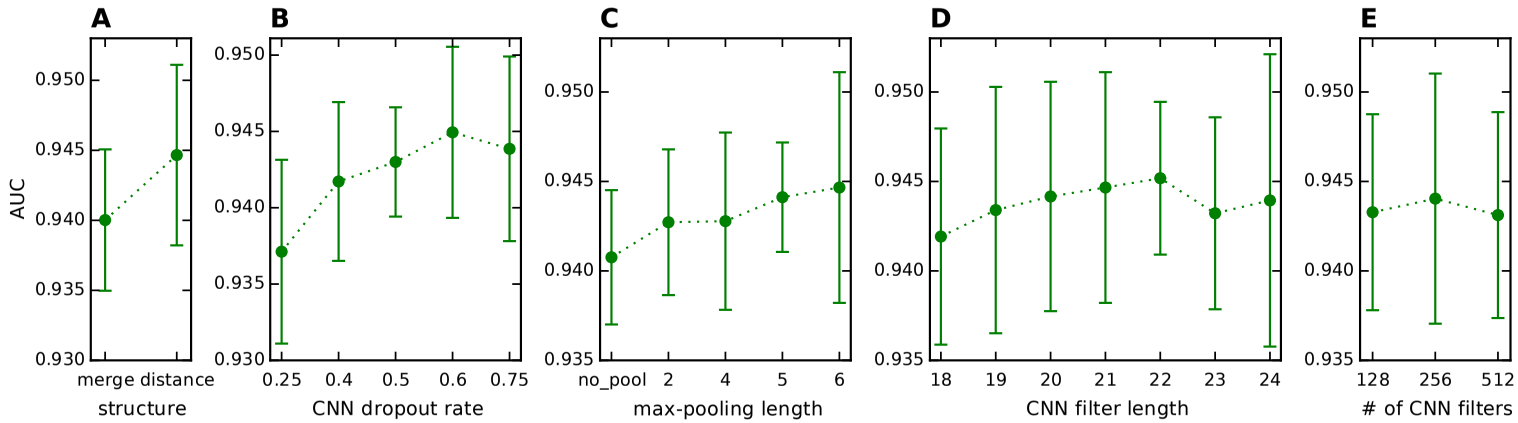

Figure 4.5: Structure and hyperparameter optimization using 5-fold cross validation tested on CCI900. (A) The structure comparison between merging and distance. The baseline hyperparameters of the CNN dropout rate, max-pooling length, CNN filter length, and number of CNN filters are 0.6, 6, 22, and 256, respectively. The performance changes as varying the (B) CNN dropout rate from 0.25 to 0.75, (C) max-pooling length from no-pool to 6, (D) CNN filter length from 18 to 24, and (E) number of CNN filters from 128 to 512.

Figure 4.5B illustrates the performance as varying with the CNN dropout rate from 0.25 to 0.75. Dropout as a regularization technique reduces overfitting by dropping out units in neural networks. In our validation experiments, a dropout rate of 0.6, meaning that 60% of units dropped out, showed the best performance and the lowest standard deviation.

The pooling layer summarized the adjacent features resulting in abstract representations, and inputs were down-sampled, resulting in a smaller number of model parameters to learn [5]. A larger pooling length means a greater summary effect. In our experiments, the performances of non-pooling layers and pooling layers with pool-lengths from 2 to 6 in terms of AUC showed in Figure 4.5C. To achieve better summarization effects and a better performance, we used max-pooling with a length of 6 as the baseline parameter.

The key features of CNNs are parameter sharing architecture and translation invariance characteristics, which are due to the use of filters (or kernels). The filter length depends on the characteristics of patterns to discover. The number of filters controls the capacity, depending on the complexity of the task. As shown in Figure 4.5D and 4.5E, we used a filter length of 22 and 256 filters as the baseline parameters. A filter length of 22 showed the highest AUC. The relatively longer molecule patterns compared to those of the fingerprints can be captured with these CNN filters.

4.2.3 Performance Comparison with Other Methods

We compared our proposed method, DeepCCI, with a feedforward neural network (FFNN), which is a plain neural network classifier described in Figure 4.1A, and the conventional machine learning methods of the support vector machine (SVM) [108, 109], random forest (RF) [10, 98], and adaptive boosting (AdaBoost) [31].

Table 4.2: Details of experimental results on CCI900 in terms of eight different evaluation metrics. The experiments were carried out with thirteen method-representation combinations. All values were the average of five repeated experiments. The best performance for each evaluation metrics and each dataset was shown with bold

data	R*	methods	AUC	ACC	MCC	TPR	TNR	PPV	NPV	F1
CCI900	S†	**DeepCCI**	**0.9537**	**0.8895**	**0.7851**	0.8347	**0.9471**	**0.9433**	0.8450	**0.8856**
	PubChem	FFNN	0.9231	0.8536	0.7086	0.8296	0.8787	0.8781	0.8308	0.8531
		SVM	0.8471	0.7606	0.5240	0.7231	0.8000	0.7918	0.7330	0.7559
		RF	0.9258	0.8380	0.6760	0.8572	0.8179	0.8320	0.8448	0.8444
		Ada	0.7908	0.7214	0.4457	0.6826	0.7622	0.7512	0.6954	0.7153
	ECFP	FFNN	0.9392	0.8692	0.7392	0.8626	0.8762	0.8807	0.8590	0.8711
		SVM	0.8944	0.8066	0.6155	0.7740	0.8409	0.8366	0.7796	0.8040
		RF	0.9439	0.8690	0.7385	0.8590	0.8797	0.8825	0.8557	0.8706
		Ada	0.8041	0.7276	0.4612	0.6662	0.7921	0.7712	0.6928	0.7149
	MACCS	FFNN	0.9249	0.8626	0.7258	0.8551	0.8705	0.8745	0.8515	0.8645
		SVM	0.7947	0.7214	0.4444	0.6961	0.7480	0.7440	0.7006	0.7193
		RF	0.9396	0.8628	0.7258	**0.8865**	0.8378	0.8519	**0.8754**	0.8688
		Ada	0.7697	0.6969	0.3944	0.6826	0.7118	0.7136	0.6807	0.6978

* Representation type of chemical compound
† SMILES string

Table 4.3: Details of experimental results on CCI800 in terms of eight different evaluation metrics. The experiments were carried out with thirteen method-representation combinations. All values were the average of five repeated experiments. The best performance for each evaluation metrics and each dataset was shown with bold

data	R*	methods	AUC	ACC	MCC	TPR	TNR	PPV	NPV	F1
CCI800	S†	**DeepCCI**	**0.9844**	**0.9516**	**0.9033**	0.9416	**0.9614**	**0.9604**	0.9432	**0.9509**
	PubChem	FFNN	0.9738	0.9253	0.8507	0.9328	0.9178	0.9185	0.9324	0.9256
		SVM	0.8713	0.7901	0.5804	0.7716	0.8084	0.7998	0.7811	0.7854
		RF	0.9695	0.9095	0.8193	0.9235	0.8956	0.8977	0.9219	0.9104
		Ada	0.8184	0.7432	0.4870	0.7167	0.7695	0.7552	0.7325	0.7354
	ECFP	FFNN	0.9821	0.9438	0.8877	**0.9496**	0.9381	0.9383	**0.9494**	0.9439
		SVM	0.9192	0.8405	0.6811	0.8357	0.8454	0.8428	0.8383	0.8392
		RF	0.9806	0.9379	0.8757	0.9352	0.9405	0.9397	0.9360	0.9375
		Ada	0.8297	0.7477	0.4983	0.6913	0.8037	0.7774	0.7241	0.7318
	MACCS	FFNN	0.9721	0.9224	0.8450	0.9258	0.9191	0.9191	0.9259	0.9224
		SVM	0.8231	0.7549	0.5099	0.7380	0.7715	0.7622	0.7481	0.7499
		RF	0.9713	0.9135	0.8273	0.9276	0.8995	0.9015	0.9261	0.9144
		Ada	0.8033	0.7340	0.4682	0.7171	0.7508	0.7406	0.7279	0.7286

* Representation type of chemical compound
† SMILES string

Table 4.4: Details of experimental results on CCI700 in terms of eight different evaluation metrics. The experiments were carried out with thirteen method-representation combinations. All values were the average of five repeated experiments. The best performance for each evaluation metrics and each dataset was shown with bold

data	R*	methods	AUC	ACC	MCC	TPR	TNR	PPV	NPV	F1
CCI700	S†	**DeepCCI**	**0.9911**	**0.9624**	**0.9247**	0.9618	**0.9630**	**0.9629**	0.9618	**0.9623**
	PubChem	FFNN	0.9857	0.9438	0.8877	0.9481	0.9395	0.9400	0.9477	0.9441
		SVM	0.8747	0.7936	0.5874	0.7797	0.8074	0.8018	0.7857	0.7906
		RF	0.9800	0.9315	0.8631	0.9420	0.9208	0.9225	0.9408	0.9322
		Ada	0.8180	0.7406	0.4819	0.7158	0.7654	0.7531	0.7294	0.7340
	ECFP	FFNN	0.9901	0.9597	0.9194	**0.9665**	0.9528	0.9534	**0.9661**	0.9599
		SVM	0.9257	0.8479	0.6958	0.8434	0.8523	0.8509	0.8449	0.8472
		RF	0.9884	0.9529	0.9059	0.9552	0.9507	0.9509	0.9550	0.9530
		Ada	0.8302	0.7488	0.4997	0.7015	0.7960	0.7746	0.7274	0.7362
	MACCS	FFNN	0.9840	0.9409	0.8818	0.9437	0.9381	0.9384	0.9434	0.9410
		SVM	0.8272	0.7555	0.5112	0.7402	0.7708	0.7635	0.7480	0.7516
		RF	0.9805	0.9329	0.8662	0.9463	0.9196	0.9217	0.9449	0.9338
		Ada	0.8113	0.7347	0.4697	0.7158	0.7536	0.7438	0.7263	0.7295

* Representation type of chemical compound
† SMILES string

The hyperparameters of SVM, RF, and AdaBoost were tested with grid search. We used the penalty parameter of 0.1 for SVM, and 100 as the number of estimators for RF and AdaBoost. Generally, the performance of the ensemble method, such as RF and AdaBoost, increases as the number of estimators increases and converges to a certain performance as the number of estimators increases. Considering the computational resources and performance, we set the number of estimators to 100.

The hyperparameters of FFNN, which is a neural network of three fully-connected layers were configured with 1,024 and 128 hidden units, ReLU for the hidden activation function, sigmoid for the output activation function, and a dropout rate of 0.25.

Table 4.2, 4.3, and 4.4 showed the overall detailed experimental results in terms of eight evaluation metrics. All the values were averaged through five repeated experiments and showed to be stable with a standard deviation of less than 0.004 in terms of AUC for all datasets. In most of the metrics except TPR and NPV, and all datasets, DeepCCI showed the best performance, followed by FFNN and RF. AdaBoost and SVM showed unsatisfactory results, giving an MCC of under 0.7 for all datasets and fingerprints.

In terms of AUC, Figure 4.6 shows the visualized performance with respect to the method, representation type, and dataset. ROC curves of DeepCCI and ECFP-based methods from one of the test results are shown in Figure 4.6A. The overall AUC results with respect to the methods are shown in Figure 4.6B. DeepCCI shows the widest area under the curve (AUC) followed by FFNN and RF, and shows a big difference with SVM and AdaBoost; DeepCCI, FFNN, SVM, RF, and AdaBoost showed mean AUC values of 0.98, 0.96, 0.86, 0.96, and 0.80, respectively. According to the results with respect to the representation type, SMILES shows the best performance followed by ECFP, PubChem, and MACCS; SMILES, PubChem, ECFP, and MACCS showed mean AUC values of 0.98, 0.90, 0.92, and 0.88, respectively. According to

the results with respect to the dataset, the overall performance gradually increased as the dataset size increased; the mean AUC improved from 0.88 to 0.92.

Figure 4.7 shows the visualized performance in terms of AUC, ACC, and MCC with bar plots of DeepCCI and ECFP-based methods. ECFP shows the best performance among the three fingerprints, as represented in Figure 4.6C. DeepCCIshowed the best performance followed by FFNN and RF in all datasets and in ECFP fingerprint in terms of AUC, ACC, and MCC.

According to the results, the performance differences between DeepCCI and FFNN could be interpreted as a result of the difference between the extracted features from the SMILES string and the crafted features from the fingerprint. Potential key features are automatically extracted from SMILES, but the features may not be known to domain experts and may not be embedded in the crafted fingerprints yet.

Figure 4.8 shows the experimental validation of the commutative property trained with CCI900 training data (order of A followed by B (A,B)) and tested with the switched order CCI900 training data (B followed by A (B,A)). DeepCCI showed the same training and test accuracy, on account of the effect of guaranteeing the commutative property. However, the test accuracy of FFNN, SVM, RF, and AdaBoost showed performance degradation by 18%, 15%, 19%, and 7% compared to the training accuracy. These results mean that the four methods recognized $\mathcal{I}(A, B)$ and $\mathcal{I}(B, A)$ independently and produced different prediction probabilities for them, whereas DeepCCI recognized them as the same.

4.2.4 Analysis on Frequent SMILES Characters

Language modeling generally carries out the preprocess of limiting vocabulary to the most common words; removing infrequent words. This is because it is hard to deal with all the words in the world, and having a huge vocabulary can cause an increase

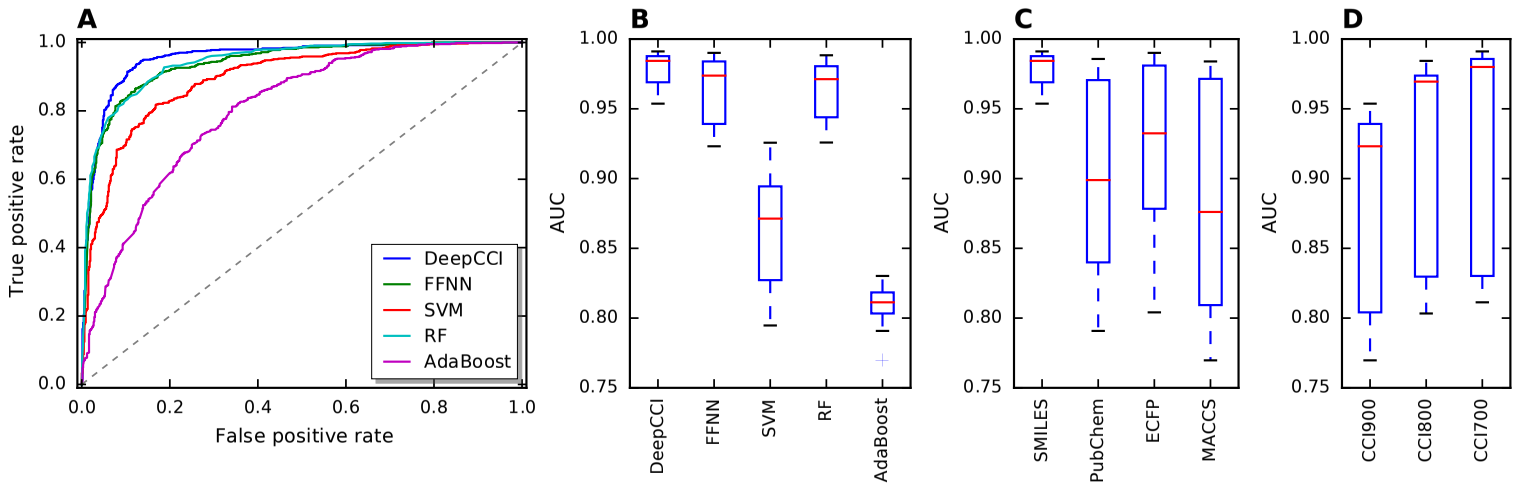

Figure 4.6: Visualized performance comparison **(A)** ROC curve of DeepCCIand other methods with ECFP, and AUC results with respect to the **(B)** each method, **(C)** each representation type, and **(D)** each dataset.

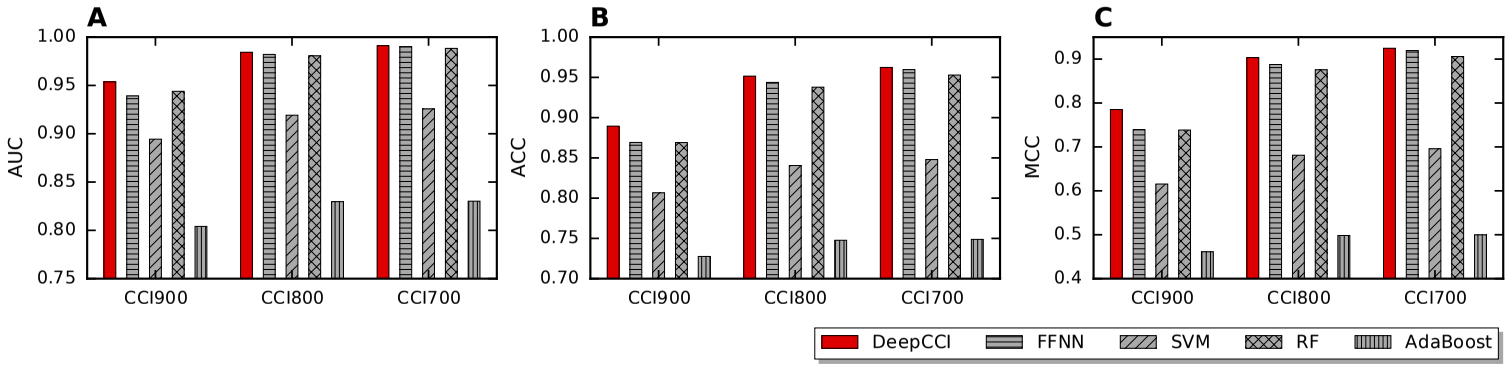

Figure 4.7: Visualized performance comparison for CCI900, CCI800, and CCI700 in terms of **(A)** AUC, **(B)** ACC, and **(C)** MCC.

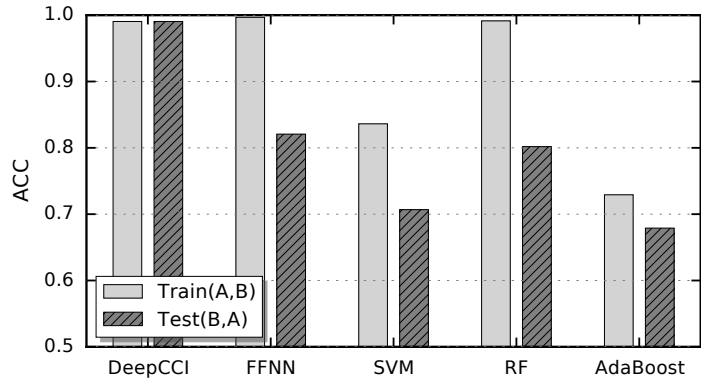

Figure 4.8: Experimental validation of the commutative property. Training accuracy for $\mathcal{I}(A, B)$ and test accuracy for the switched order $\mathcal{I}(B, A)$.

in the number of parameters to be learned and training time. Since the distribution of each character in SMILES is quite different, and the character set plays a similar role to the vocabulary in language modeling, we analyzed the effects of character sets and distribution according to the character frequency as shown in Figure 4.9.

The original character set, $\mathcal{X}$, consists of 65-character. According to the character frequency, we removed the infrequent characters in the character set and assigned them to `OTHERS`. The number of frequent characters of 10 means that top 10 frequent characters and `OTHERS` are included in the character set, $\mathcal{X}$, ($|\mathcal{X}| = 11$).

Figure 4.9A shows the performance changes according to the frequent character set. The frequent characters of top 10, not the total character set, shows the best performance. This point can be considered to be the minimized character set while covering most frequent characters. The most frequent character, `C`, accounts for 37% and the top 10 characters account for more than 95% of the overall frequency distribution. The total frequency of the remaining 55 characters is less than 5% (see Figure 4.9C).

The size of character set affects the number of parameters to be learned. Particularly in the one-hot encoding scheme, the size of character set is directly associated

with the input dimension. Figure 4.9B shows the number of parameters in the shared CNNs. As the number of frequent characters decreases, the number of parameters to learn decreases, and the decreased number of parameters can lead to a reduction in training time and demanded memory size. Considering the number of parameters, training time, memory, and performance, it seems necessary to select a suitably reduced SMILES character set.

4.3 Discussion

In this paper, we proposed an end-to-end representation learning method for CCI. It showed the best performance when compared to a plain neural network classifier and conventional machine learning methods.

Deep neural network based learning has shown remarkable success in chemical analyses and attracted considerable research attention. Despite the success and interest, the crafted features designed by domain experts continue to be used, even in state-of-the-art neural network learning approaches. The main objective of deep neural network learning is not merely classification or regression from the crafted features, but rather to automatically extract meaningful features from the original input and then analyze them. For this reason, we have strived to perform end-to-end representation learning based on SMILES. Our approach showed better results than using the crafted fingerprints. With the CNN filters used in our experiments, we can capture relatively longer patterns that might otherwise be hard to discover with manually crafted fingerprints. We believe that patterns longer than 10 characters as well as short and simple patterns in a SMILES string play a particular role in the biological reaction between chemicals. There are a variety of well-refined chemical features, and PubChem, ECFP, and MACCS fingerprints are the competent and widely used

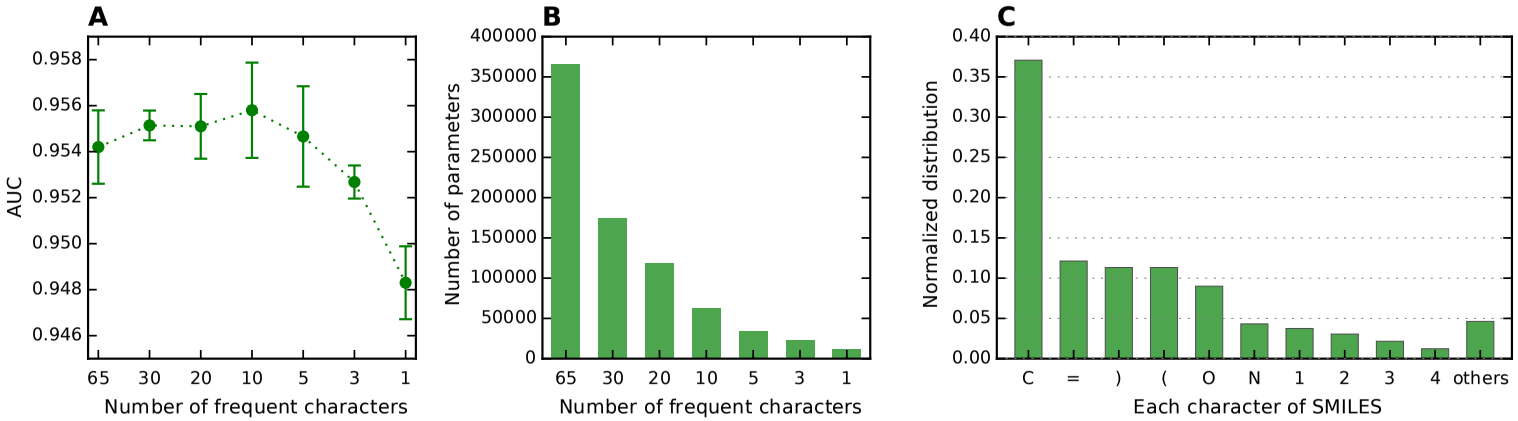

Figure 4.9: Effects of SMIILES input characters. (A) Performance evaluation in terms of AUC and (B) the number of parameters in shared CNN networks according to the number of frequent SMILES characters. (C) Normalized distribution of top 10 SMILES character and rest.

descriptors. Through a comparison with popularly used three fingerprints, we confirmed the possibility of end-to-end SMILES learning.

Automatic feature learning alleviates the sophisticated efforts required for feature engineering of domain experts and it can enable discovery of meaningful features that are possibly not known to domain experts. We expect that the proposed end-to-end SMILES learning will be applied to other drug (chemical) analyses, such as QSAR prediction, chemical toxicology, and drug target prediction, and it will improve prediction performance.

In addition to the end-to-end SMILES learning framework, we guaranteed the commutative property in the prediction of CCI. Since CCI is a homogeneous interaction problem, it should produce the same interaction probability for $\mathcal{I}(A, B)$ and its converse, $\mathcal{I}(B, A)$. To guarantee the commutative property we used the shared representation extracting network and L1 element-wise distance techniques based on Siamese architecture, and confirmed the property with experimental verification. We expect this architecture can be applied to other homogeneous symmetric problems, such as protein-protein and gene-gene interactions, as well as other homogeneous comparisons.

The future revisions of DeepCCI may include the following improvements: We are planning to analyze how the features automatically extracted from shared CNNs affect the overall performance when they are applied to the conventional machine learning classifiers. As in other fingerprint-based approaches, deep learning-based hidden features may be used as a separate fingerprint for various classifiers. Instead of zero-padding, we may consider using a wildcard character in encoding as a special character, as was done in the idea of wildcard denoising [93]. All the zero values included in one-hot encoding may confuse the network, making it fail to recognize them as a padded region.

Chapter 5

Comprehensive Ensemble with CNN, RNN Based Learner

Quantitative structure activity relationship (QSAR) [73] modeling is to reveal the relationship between biological activity and chemical structural properties. With the relationship modeling, we can reduce candidate chemicals enormously or prioritize a large number of chemicals, related to a particular certain biological activity. Reducing or prioritizing candidate chemicals with trusty is directly related to cost saving because experiment works for the validation process in drug development are costly greatly.

QSAR modeling (prediction) is a very important and inevitable process in the pharmaceutical industry, but the prediction has many constraints; QSAR data may involve a very large number of chemicals (more than hundred thousands); each chemical can be expressed with many kinds of diverse descriptors (more than thousands); commonly used fingerprints are very sparse (most of the values are zero) and some values are in a highly correlated; the true labels of the chemicals are not highly trusted, should be assumed to have noise.

With these constraints, QSAR modeling is a very difficult problem and trusted predictions are always demanded. In conventional machine learning approaches, random forest (RF) are most commonly used with high level of prediction performance. RF is a kind of ensemble method based on multiple decision trees and it avoids the overfitting from a single decision tree. Most QSAR prediction methods compare their performance to RF.

The Merck Kaggle competition in 2012 turns people's attention into neural networks. The winning team used multi-task neural networks (MTNN). Fundamental learning structure is based on plain feed forward neural networks but avoids overfitting by learning multiple bioassays concurrently. The team showed outperforming results to RF consistently. Although the major contributor of such a high performance was the technique of multi-task neural network, the team used a method ensemble (ensemble between different learning methods) for a final decision to get better prediction performance.

Both random forest and the winning team of Kaggle competition are based on the ensemble learning technique. Theoretically and empirically, it has been represented that a prediction power of ensemble learning exceeds that of single (individual) learner. Ensemble learning manages the strengths and weakness of single learners, just like people often consider diverse opinions when faced with very important problems.

In addition to the two methods, there have been proposed various ensemble methods in drug (chemical) research; neural network ensemble against bootstrap sampling in QSAR and QSPR [1]; ensemble against different learning method for drug-drug interaction [102]; ensemble against diverse of chemical representation for carcinogenicity prediction [120]. The first two studies are based on common ensemble approaches; ensemble against data sampling (bagging), ensemble against methods. The

third one is a drug-specific ensemble technique; ensemble against input representation. However, these ensemble approaches limit ensemble to a certain subject.

We propose a multi-subject comprehensive ensemble with a new type of individual classifier. More detailed description of our proposed methods are as follows;

- Instead of limiting ensemble to a certain subject, we ensemble every combination of individual methods comprehensively; ensemble against the combinations of bagging, methods, and representations.
- We propose a new type of QSAR individual classifier, which is an end-to-end neural network model based on one-dimensional convolutional neural networks (CNNs) and recurrent neural networks (RNNs).
- We combine the multiple individual methods by using second-level learning, called stacking; we can give optimized weight on each first-level model through learning.

We confirmed the superiority of our proposed methods by comparing with individual classifiers, and other combining technique, and the performance of MTNN represented in Dahl et al. [20].

5.1 Background

5.1.1 Ensemble Learning

Ensemble learning builds a predictive model by combining multiple constituent learning algorithms. Theoretically and empirically, numerous studies have demonstrated that ensemble models usually give higher accuracy than individual models. A collection of weak inducers can make a single strong ensemble model.

Framework

Ensemble learning is divided into *independent* and *dependent* frameworks for building of ensembles. Independent framework, also called randomization-based approach, the individual inducers can be trained independently and so in parallel. On the other hand, dependent framework, also called boosting-based approach, the base inducers are affected sequentially from the previous inducers. In a level of individual model, we used both the independent method, *e.g.*, random forest, and the dependent method, *e.g.*, gradient boosting. In combining level, we treated the individual models independently.

Diversity

Diversity is well known as a crucial condition for ensemble learning [54, 105]. Diversity leads to uncorrelated inducers, which in turn improve the final performance [41]. We focused on the three types of diversity.

- *Diversity on training data with sampling*

  Original dataset can be diversified by random sampling with replacement (bootstrapping). By sampling N number of times independently and randomly, we can generate m number of new datasets, D_i (for i is $1..m$), of the same N size with the original dataset, D. By sampling with replacement, the new dataset consists of duplicate and unique ($\approx 63.2\%$ for large N) samples from the original data, and the new dataset will differ from the original in the end.

- *Diversity on learning methods (inducers)*

  Different kinds of learning algorithms (multi-inducers) produce diverse predictions, leading to improved performance that is hard to achieve with individual method alone. We used learning methods, such as random forest (RF) [10,

98], support vector machine (SVM) [108, 109], gradient boosting machine (GBM) [32], and neural network (NN). We compared the individual predictions with combined ensemble results in our experiments.

- *Diversity on input representations*

  Unlike ordinary problems, drug (chemical) can be expressed with diverse of representations. Zhang et al. [120] showed improved performance through ensemble models from multiple molecular fingerprints. We used PubChem [110], ECFP [71], MACCS [25] fingerprints and a simplified molecular input line entry system (SMILES) [111].

Combining a Set of Models

There exist numerous methods for combining a set of models. We tested the simplest *uniform averaging*, *bayesian optimal* and combining through *second-level meta-learning* method. We have N number of individual learning models, $\mathcal{L}_i$, where $i = \{1, \cdots, N\}$, and i-th model outputs the prediction probability, P_i for give $\mathbf{x}$. By weighting, w_i, to each models, we can make the final prediction, $\hat{y} = \sum_{i=1}^{N} w_i \mathcal{L}_i(\mathbf{x})$, where if the weight $w_i = 1/N, \forall i$, it means the *uniform averaging*. *Bayesian optimal* makes the final decision through the product of probabilities, $\hat{y} = \prod_{i=1}^{N} P_i/(1 - P_i)$. Another method, we can combining the first-level output probability through *second-level meta-learning*, $\hat{y} = \boldsymbol{\mathcal{L}}(\mathcal{L}_1(\mathbf{x}), \mathcal{L}_2(\mathbf{x}), \cdots, \mathcal{L}_N(\mathbf{x}))$.

5.2 Methods

Our proposed ensemble learning procedure is divided into two stage. The first-level learning is about individual learning from different learning algorithms and different chemical representations. The prediction probabilities from these first-level learning

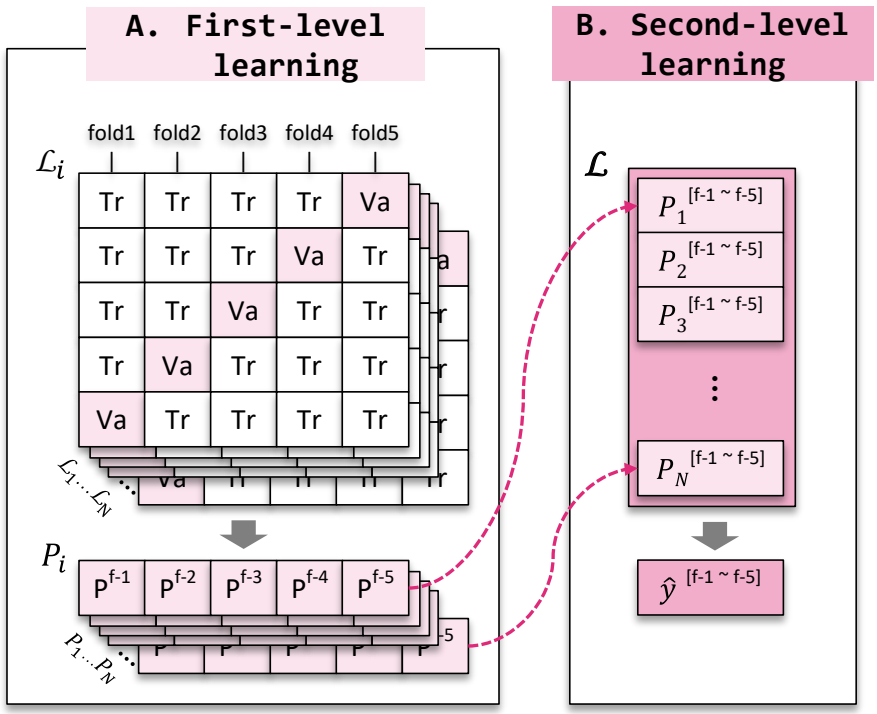

Figure 5.1: Learning procedure of the proposed method. The proposed learning procedure consists of two levels: the first-level of individual learning and the second-level of combining learning. The individual i-th learning algorithm, $\mathcal{L}_i$ outputs its own prediction probability, P_i through 5-fold cross validation. The N number of diverse learning models produces N number of prediction probabilities (P_1, P_2,$\cdots$,P_N), respectively. The N number of probabilities are concatenated and then used as input for the second-level learning. Finally, the second-level learning algorithm, $\boldsymbol{\mathcal{L}}$, makes a final decision, $\hat{y}$.

models are used as inputs for the second-level learning. The second-level learning makes the final decision from the first-level predictions, by imposing the learned importance to the individual models (see Figure 5.1).

5.2.1 Notations

The notations used in our paper are as follows:

- $\mathbf{x}$: preprocessed input
- $\mathbf{h}$: hidden representation
- $\mathcal{L}$: first-level individual learning algorithm ($\mathcal{L}_i$: i-th algorithm, $i = \{1, \cdots, N\}$)
- $\boldsymbol{\mathcal{L}}$: second-level learning algorithm
- P: predicted probability from the individual model (P_i: predicted probability from the $\mathcal{L}_i$)
- $\hat{y}$: final predicted decision from the second-level learning
- σ: activation function (σ_s: sigmoid, σ_r: rectified linear unit (ReLU), σ_t: hyperbolic tangent)
- N: total number of individual algorithms

5.2.2 First-Level: Individual Learning

With the combination of learning algorithms and chemical input representations, we generated thirteen kinds of diverse individual learning models; nine models from conventional machine learning methods, three models from the plain feed-forward neural network, and one model from the 1D-CNN and RNN based neural network.

Conventional Machine Learning Methods

We used the linear SVM, RF, and GBM from scikit-learn library. We decided the hyperparameters of penalty parameter, 0.05, for SVM and the number of estimators, 100, for RF and GBM, based on grid search and experimental efficiency. The best parameter for the number of estimators of RF was $1,000$ in 5-fold cross validation, but the average AUC difference between 100 and $1,000$ is negligible; the 100 estimator showed 0.860 and the $1,000$ estimator showed 0.864 average AUC. All the other parameters were set as default. The prediction probabilities from these learning methods were used as inputs for the second-level learning. SVM makes outputs a signed distance to the hyperplane, not a probability, so we used a probability calibration method with sigmoid for probabilistic outputs.

Plain Feed Forward Neural Network

We used a plain feed forward neural network (FFNN) for vector type fingerprints, and the neural network consisted of three fully connected layers (Fcl) of 512, 64, and 1 units for each layer, with Relu, tanh, and the last sigmoid activation function, respectively,

$$P = \sigma_s(\mathbf{Fcl}(\sigma_t(\mathbf{Fcl}(\sigma_r(\mathbf{Fcl}(\mathbf{x})))))). \tag{5.1}$$

The last sigmoid activation function outputs probability for binary classification. We used Adam optimizer [48] with binary cross-entropy loss (learning rate: 0.001, epoch: 30, and mini-batch size: 256).

Convolutional and Recurrent Neural Network

To learn from the string type of SMILES inputs, we used famous neural network structures of 1-dimensional convolutional neural networks (1D-CNNs) and recurrent

neural network (RNNs). CNNs was used to recognize local patterns (to learn the short-term dependency), and RNN used as the next layers to learn long term dependency from the recognized local patterns.

The input SMILES was preprocessed with one-hot encoding and truncated/padded to maximum length to 100. We only consider the most frequent nine characters in SMILES, thereby the encoding dimension became 9. The preprocessed input, $\mathbf{x}$, is expressed as $\texttt{sequence_length} \times \texttt{encoding_dimension}$ vector. The preprocessed $\mathbf{x}$ fed into CNN layers without pooling layer (CNN filer length: 17, the number of filters: 384). Then, the output of CNN fed into GRU layer (dimension: 9, framework: many-to-many RNN structure).

$$\mathbf{h} = \sigma_t(\mathbf{GRU}(\sigma_r(\mathbf{Conv}(\mathbf{x})))) \tag{5.2}$$

where $\mathbf{h}$ is the output of GRU and CNN layers, σ_r is the ReLU, σ_t is the hyperbolic tangent. The $\mathbf{h}$ is flattened and then fed into the fully-connected neural network.

$$P = \sigma_s(\mathbf{Fcl}(\sigma_r(\mathbf{Fcl}(\mathbf{h}_{\texttt{flatten}})))) \tag{5.3}$$

where P is the output probability from the last sigmoid activation function for binary classification (active vs inactive).

When training we used dropout for each layer (in CNN: 0.9, RNN: 0.6, the first Fcl: 0.6) and Adam optimizer (learning rate: 0.001, epoch: 120, mini-batch size: 256) with the binary cross-entropy. Most of these hyperparameters were empirically determined.

5.2.3 Second-Level: Meta-Learning

For the final decision, we combine the first-level predictions which are generated from the set of individual models. The performance of individual methods varies depending on each dataset as represented in Table 5.2; there is no invincible universal method. Combining weights on learning methods should be applied differently according to the dataset. We exploit the learning based combining methods called *stacking* rather than just simple averaging or voting.

Stacking is an ensemble combining techniques, which tries to make a final decision from the predictions generated by first-level learning algorithms.

$$\hat{y} = \boldsymbol{\mathcal{L}}\left([P_1, P_2, \cdots, P_n]\right) \tag{5.4}$$

where, $\boldsymbol{\mathcal{L}}$ is a second-level learning algorithm, and any machine learning methods could be used as the *stacking* learner. The $\hat{y}$ is the final decision and P_i is a predicted probability of the first-level model, $\mathcal{L}_i$. All the $P_i, i = \{1, 2, \cdots, N\}$ are concatenated and used as inputs. The model importance imposing weight on P_i is determined through learning.

5.3 Results

5.3.1 Experimental Setup

Dataset

Bioassay is a biochemical test to determine or estimate potency of a chemical compound on subjects and has been used for a variety of purposes, including drug development, toxicity analysis, and environmental impact analysis. In our experiment, we used nineteen bioassays downloaded from PubChem open chemistry database [110]

Table 5.1: Details of the bioassay datasets used in the experiments. The nineteen bioassays are identical to those specified in Dahl et al. [20]

Assay ID	Description of BioAssay	# Active	# Inactive	Source	Ver
1851_1a2	Cytochrome P450 Panel Assay, cyp1a2	5,902	6,974	NCGC*	1.2
1851_2c19	Cytochrome P450 Panel Assay, cyp2c19	5,840	7,135	NCGC	1.2
1851_2c9	Cytochrome P450 Panel Assay, cyp2c9	4,065	8,361	NCGC	1.2
1851_2d6	Cytochrome P450 Panel Assay, cyp2d6	2,601	10,826	NCGC	1.2
1851_3a4	Cytochrome P450 Panel Assay, cyp3a4	5,175	7,446	NCGC	1.2
1915	Streptokinase Expression Inhibition	2,219	1,017	Broad Institute	1.2
2358	Inhibitors of Protein Phosphatase 1 (PP1)	1,006	934	Scripps†	1.2
463213	Inhibitors of tim10-1 yeast	4,138	3,234	Burnham‡	1.3
463215	Inhibitors of tim10 yeast	2,941	1,695	Burnham	1.2
488912	Inhibitors of Sentrin-specific protease 8 (SENP8)	2,491	3,705	Burnham	1.3
488915	Inhibitors of Sentrin-specific protease 6 (SENP6)	3,568	2,628	Burnham	1.3
488917	Inhibitors of Sentrin-specific protease 7 (SENP7)	4,283	1,913	Burnham	1.1
488918	Inhibitors of Sentrin-specific proteases (SENPs)	3,691	2,505	Burnham	1.2
492992	Inhibitors of the two-pore domain potassium channel (KCNK9)	2,094	2,820	Johns Hopkins§	1.2
504607	Inhibitors of Mdm2/MdmX interaction	4,825	1,406	Burnham	1.1
624504	Inhibitors of the mitochondrial permeability transition pore	3,944	1,090	Burnham	1.1
651739	Inhibition of T.cruzi proliferation	4,043	1,322	Broad Institute	1.1
651744	NIH/3T3 (mouse embryonic fibroblast) toxicity	3,099	2,303	Broad Institute	1.1
652065	Molecules that bind r(CAG) RNA repeats	2,965	1,286	Scripps	1.1

* The NCATS Chemical Genomics Center, formerly kwown as NIH Chemical Genomics Center

† The Scripps Research Institute Molecular Screening Center

‡ Burnham Center for Chemical Genomics

§ Johns Hopkins Ion Channel Center

as listed in Table 5.1. All the bioassays are identical to those specified in Dahl et al. [20]. The purpose of the paper was to address the multi-task effects, and thus there exist closely related experimental assays, such as 1851, 46321*, 48891*, and 6517** series.

From each bioassay, we extracted PubChem chemical ID and activity outcome of active or inactive. The duplicate chemicals were used only once, and conflicting chemicals that have both active and inactive outcomes were excluded. The number of active and inactive preprocessed chemical IDs is listed in Table 5.1. The class ratio between active and inactive are diverse ranging from 1:1.1 to 1:4.2, but most of the bioassays are imbalanced with an average ratio of 1:2.

Representation of Chemical Compound

The chemical compounds can be expressed with various types of chemical descriptors representing structural information. In our experiment, we used one string type of SMILES [111] and three binary vector type fingerprints of PubChem [110], ECFP [71], and MACCS [25]. Since the SMILES is a sequential string type of descriptor, it is not a proper form for the conventional learning methods. We used an end-to-end neural network structure capable of handling sequential form, such as 1D-CNN or RNN for SMILES. On the other hand, the binary vector type fingerprints consist of 1 and 0 independently, we used the conventional machine learning approaches and general feed-forward neural network.

The SMILES and PubChem fingerprint were retrieved from the preprocessed chemical IDs by using PubChemPy [99], and ECFP and MACCS fingerprints were retrieved from the SMILES by using RDKit [56]. An example of SMILES and vector values of Leucine, which is an essential amino acid for hemoglobin formation, is as follow:

- PubChem chemical ID: 6106
- SMILES string: `CC(C)CC(C(=O)O)N`
- PubChem fingerprint: $1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, \cdots$
- ECFP fingerprint: $0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \cdots$
- MACCS fingerprint: $0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \cdots$
  (Most features of fingerprints are very sparse (mostly zeros).)

Experimental Configuration and Environment

We divided the dataset into two parts randomly: 75% of the dataset used as training set and the other 25% used as testing set. The training dataset also randomly partitioned into five; one partition used as validation and the rest four partitions used as training (5-fold cross validation). The predicted probability, $P^{\texttt{fold i}}$, for each fold-i validation set was concatenated, P, and then used as inputs for the second-level learning. The second-level learning algorithm, $\boldsymbol{\mathcal{L}}$, trained with set of $\mathbf{P} = [P_1, P_2, \cdots, P_N]$ generated from diverse individual models.

We run our experiments on Ubuntu 14.04 (3.5GHz Intel i7-5930K and GTX Titan X Maxwell(12GB)). We used Keras library package (version 2.0.6) for neural network implementation, Scikit-learn library package (version 0.18) for conventional machine learning methods, and PubChemPy (version 1.0.3) and RDKit (version 1.0.3) used for input representation preparation of chemical compound.

5.3.2 Performance Comparison with Other Models

We compared our proposed methods with a total of thirteen models: twelve models from the combination of three types of fingerprints (PubChem, ECFP, and MACCS)

Table 5.2: Performance comparison the proposed method with single models. Each value is the averaged AUC of five repeated experiments (bold: top 3)

BioAssay	PubChem fingerprint				ECFP fingerprint				MACCS fingerprint				SMILES	proposed
	RF	SVM	GBM	NN	RF	SVM	GBM	NN	RF	SVM	GBM	NN	NN	
1851_1a2	**0.921**	0.896	0.900	0.921	0.919	0.909	0.882	0.919	0.913	0.879	0.894	0.912	**0.922**	**0.943**
1851_2c19	0.871	0.852	0.848	0.873	**0.883**	0.876	0.854	0.879	0.874	0.842	0.850	**0.884**	0.876	**0.908**
1851_2c9	0.871	0.858	0.851	0.874	**0.881**	0.870	0.844	**0.881**	0.857	0.828	0.840	0.870	0.877	**0.908**
1851_2d6	0.858	0.847	0.832	**0.869**	**0.866**	0.855	0.834	0.853	0.856	0.815	0.830	0.852	0.846	**0.892**
1851_3a4	0.878	0.868	0.865	0.886	0.891	**0.892**	0.855	**0.895**	0.867	0.832	0.851	0.875	0.891	**0.920**
1915	**0.754**	0.694	0.709	0.718	0.736	0.719	0.700	0.710	**0.761**	0.716	0.736	0.740	0.697	**0.764**
2358	**0.786**	0.709	0.736	0.771	**0.781**	0.770	0.724	0.763	0.772	0.734	0.763	0.776	0.697	**0.807**
463213	**0.672**	0.640	0.652	0.654	**0.687**	0.658	0.645	0.663	0.667	0.641	0.655	0.652	0.634	**0.694**
463215	0.618	0.577	0.592	0.604	0.620	0.588	0.597	0.598	**0.629**	0.600	**0.630**	0.624	0.587	**0.634**
488912	**0.679**	0.642	0.634	0.667	**0.692**	0.658	0.666	0.676	0.668	0.634	0.650	0.676	0.646	**0.700**
488915	**0.717**	0.686	0.679	0.714	**0.730**	0.693	0.679	0.707	0.689	0.658	0.680	0.695	0.676	**0.739**
488917	**0.807**	0.777	0.759	0.802	**0.814**	0.792	0.760	0.798	0.787	0.726	0.752	0.787	0.780	**0.841**
488918	0.762	0.746	0.735	**0.782**	**0.777**	0.771	0.728	0.766	0.739	0.691	0.708	0.742	0.746	**0.801**
492992	**0.828**	0.783	0.783	0.804	**0.851**	0.815	0.802	0.822	0.828	0.726	0.759	0.792	0.802	**0.862**
504607	**0.695**	0.679	**0.691**	0.689	0.690	0.680	0.676	0.658	0.674	0.646	0.662	0.661	0.653	**0.726**
624504	**0.884**	0.853	0.857	0.865	**0.884**	0.869	0.857	0.862	0.875	0.833	0.862	0.876	0.868	**0.904**
651739	0.790	0.772	0.773	0.781	**0.802**	0.780	0.771	0.787	0.779	0.729	0.759	0.753	**0.792**	**0.809**
651744	0.884	0.861	0.872	0.886	0.890	0.884	0.875	**0.896**	0.868	0.828	0.843	0.850	**0.898**	**0.909**
652065	**0.801**	0.757	0.782	0.784	**0.784**	0.782	0.757	0.770	0.780	0.737	0.759	0.772	0.762	**0.832**
average	**0.793**	0.763	0.766	0.787	**0.799**	0.782	0.763	0.784	0.783	0.742	0.762	0.778	0.771	**0.821**

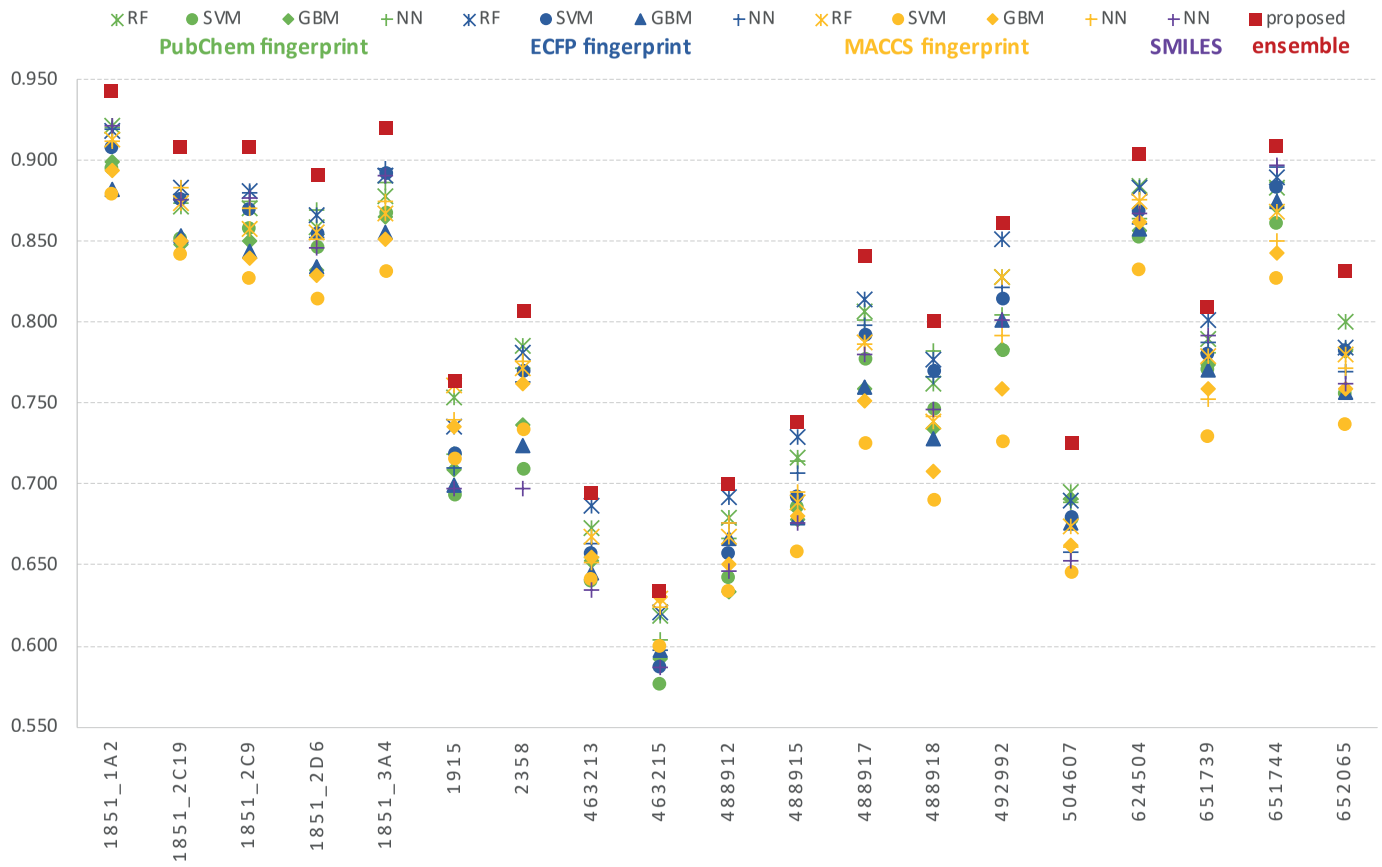

Figure 5.2: Visualized performance comparison the proposed method with other models. Each value is represented with marker style and color (best viewed in color). The classification methods are distinguished by different marker style, and the chemical descriptors (or fingerprints) are distinguished by different color. The proposed ensemble method is represented with a red square.

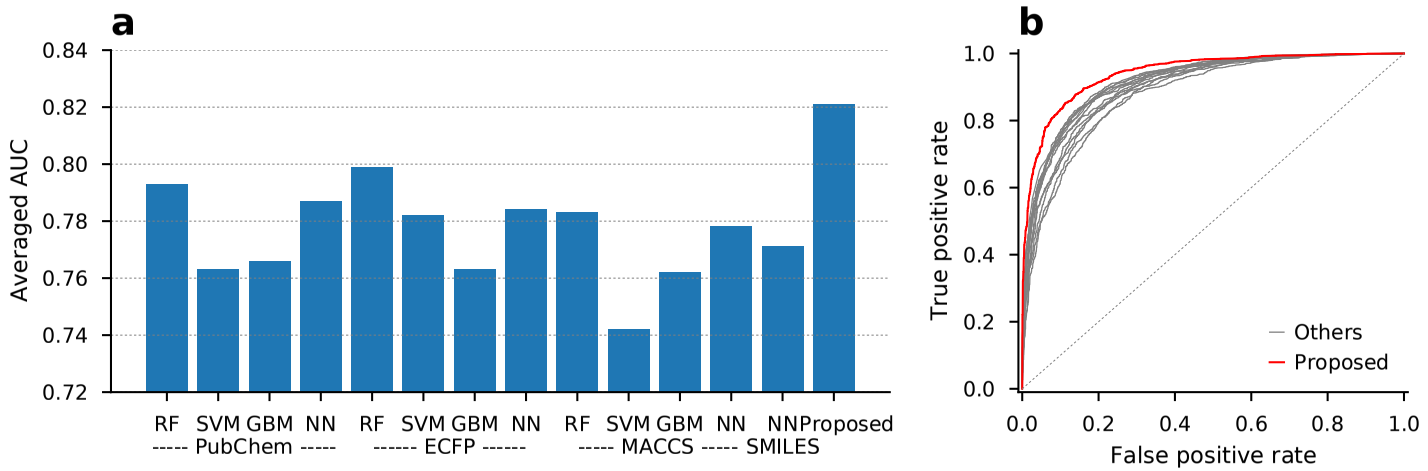

Figure 5.3: Performance comparison in terms of averaged AUC and ROC plot. **(a)** Averaged AUC over the datasets. Each AUC is the averaged over nineteen datasets. The last bar represents proposed method. **(b)** Receiver operating characteristic (ROC) plot of `1851_1a2` dataset.

and four types of learning methods (RF, SVM, GBM, and NN), and one model from SMILES-NN combination.

As represented in Table 5.2, the proposed methods showed the best performance across all the datasets. In terms of the top-3 (represented in bold), the ECFP-RF and PubChem-RF models followed. However, we could see the top-3 dispersed across chemical descriptors and learning methods except for PubChem-SVM, ECFP-GBM, and MACCS-SVM. The SMILES-NN models showed within the top-3 rank in three datasets. On the learning methodological side (in single models), RF showed the highest number of top-3 AUCs followed by NN, GBM, and SVM. On the chemical representation side, ECFP showed the highest number of top-3 AUCs followed by PubChem, SMILES (compared proportionally), and MACCS.

In comparison with results in Dahl et al. [20], the proposed ensemble method showed better performance in thirteen out of nineteen datasets; only the rest six datasets (`463215`, four `48891`$\star$ series, and `651739`) showed lower performance. Except the six datasets, the proposed method and Dahl et al. [20] showed average AUC of 0.851 and 0.828, respectively. Ensemble approach does not take into account the relevance of datasets, but predictions from multiple learners. On the other hand, multi-task learning simultaneously takes into account the relevance of dataset and learns generalized hidden representations. The relevancy between dataset could give a positive effect on multi-task learning, and the experimented datasets consisted of a series of very relevant bioassays. Even with the relatedness of the datasets, ensemble method showed higher AUCs than the multi-task approach in more datasets.

In the visualized results (Figure 5.2), the proposed method (red square) is spotted on top of every dataset, spaced apart from other results. Most of the yellow markers (MACCS) appears at the bottom of the results. MACCS-SVM showed the lowest AUC in all datasets except four datasets (`1915`, `2358`, `463213`, and `463215`). The

difference between the best-AUC (proposed) and the worst-AUC is ranging from 5.7% to 13.6%, and with an average of 8.3%.

In terms of the averaged AUC (Figure 5.3a), the proposed methods showed the best performance of 0.821, followed by ECFP-RF of 0.799 and PubChem-RF of 0.793. MACCS-SVM combination showed the lowest AUC of 0.742. Except the best (proposed) and worst (MACCS-SVM) methods, all the AUCs are ranging from 0.76 to 0.80. Prediction performance varies depending on the learning methods and input representations; while SVM showed better performance than GBM in ECFP, GBM showed better performance than SVM in MACCS. The ROC plot (Figure 5.3b) represents an example of `1851_1a2` results. The proposed curve (red) covers up all the other single methods, showing the highest AUC (0.943). Although we did not draw ROC AUC plots for all datasets, other plots showed similar patterns to Figure 5.3b.

5.3.3 Performance Comparison on Ensemble Combination

In addition to the comparison with other single models, we compared classification performance based on ensemble combination (aggregation) methods. The performance can vary depending on the combination methods between base inducers (models or classifiers). Typically, the equal-weighted voting or averaging combination methods can be easily applied; in case of bootstrap aggregation (called bagging), the base-classifiers are combined with equal weights. We compared proposed comprehensive ensembles with other partial ensemble approaches (Table 5.3). The partial ensembles are combined by average ensemble, and the comprehensive ensembles are combined by average, Bayesian optimal, and meta-learning. For the meta-learning, we compared five learning algorithms; network network (NN), SVM, RF, GBM, and linear regression (ordinary least square) (Table 5.4). We further analyzed the importance of models based on the learned weights (Figure 5.4).

As represented in Table 5.3, the first three columns represent classification per-

Table 5.3: Performance comparison on ensemble combination methods. Each value is the averaged AUC of five repeated experiments (bold: top 3). All the ensembles except the last three are based on partial averaged ensemble, the last three are comprehensive ensembles. Among comprehensive ensembles, first one is based on average, the second one is based on bayesian optimal, and the last one is based on *meta-learning* (second-level learning). The first three columns are ensemble between methods, the next five columns are ensemble between chemical representations.

BioAssay	partial averaged ensemble								comprehensive ensemble		
	Method Ensemble (ME)			Representation Ensemble (RE)							
	PubChem	ECFP	MACCS	RF	SVM	GBM	NN	NN (+SMILES)	average	bayesian	meta-learning
1851_1a2	0.924	0.924	0.915	0.931	0.917	0.907	0.937	**0.941**	0.935	**0.938**	**0.943**
1851_2c19	0.879	0.892	0.883	0.893	0.885	0.868	0.902	**0.905**	0.902	**0.904**	**0.908**
1851_2c9	0.882	0.890	0.872	0.888	0.881	0.865	0.900	**0.906**	0.900	**0.903**	**0.908**
1851_2d6	0.875	0.876	0.862	0.880	0.864	0.853	0.883	0.885	**0.886**	**0.889**	**0.892**
1851_3a4	0.892	0.905	0.879	0.898	0.897	0.881	0.912	**0.919**	0.915	**0.917**	**0.920**
1915	0.743	0.733	0.759	**0.769**	0.736	0.739	0.746	0.748	**0.763**	0.761	**0.764**
2358	0.779	0.781	0.788	0.805	0.781	0.773	**0.807**	0.805	**0.808**	**0.808**	0.807
463213	0.675	0.685	0.671	0.689	0.667	0.665	0.684	0.686	**0.693**	**0.694**	**0.694**
463215	0.614	0.613	**0.641**	**0.636**	0.605	0.622	0.623	0.624	0.632	0.633	**0.634**
488912	0.680	0.694	0.684	0.698	0.666	0.667	0.697	0.700	**0.704**	**0.706**	**0.700**
488915	0.725	0.726	0.704	0.729	0.708	0.699	0.734	**0.738**	0.737	**0.738**	**0.739**
488917	0.814	0.820	0.794	0.823	0.802	0.782	0.832	**0.838**	0.835	**0.837**	**0.841**
488918	0.783	0.784	0.749	0.780	0.777	0.752	0.793	**0.799**	0.799	**0.800**	**0.801**
492992	0.826	0.840	0.807	**0.855**	0.816	0.812	0.839	0.847	0.851	**0.852**	**0.862**
504607	0.717	0.704	0.689	0.708	0.704	0.704	0.702	0.710	**0.727**	**0.724**	**0.726**
624504	0.886	0.886	0.877	0.897	0.883	0.879	0.891	0.900	**0.900**	**0.901**	**0.904**
651739	0.802	**0.812**	0.785	0.800	0.775	0.783	0.803	0.806	**0.813**	**0.814**	0.809
651744	0.894	0.903	0.871	0.890	0.879	0.879	0.899	**0.904**	0.901	**0.905**	**0.909**
652065	0.808	0.797	0.792	0.808	0.803	0.802	0.813	0.822	**0.830**	**0.830**	**0.832**
average	0.800	0.803	0.791	0.809	0.792	0.786	0.810	**0.815**	**0.817**	**0.819**	**0.821**

Table 5.4: Performance comparison of comprehensive ensemble meta-learning methods. For the second-level meta-learning, we compare five learning methods and used the probabilities from thirteen individual learning algorithms as inputs.

BioAssay	NN	**SVM**	RF	GBM	Regression
1851_1a2	0.943	0.943	0.938	0.942	0.943
1851_2c19	0.908	0.908	0.901	0.906	0.908
1851_2c9	0.907	0.908	0.903	0.905	0.908
1851_2d6	0.893	0.892	0.879	0.890	0.892
1851_3a4	0.920	0.920	0.914	0.918	0.920
1915	0.766	0.764	0.745	0.755	0.764
2358	0.807	0.807	0.794	0.794	0.800
463213	0.695	0.694	0.681	0.690	0.693
463215	0.634	0.634	0.603	0.610	0.633
488912	0.702	0.700	0.687	0.693	0.698
488915	0.739	0.739	0.728	0.731	0.736
488917	0.838	0.841	0.827	0.832	0.840
488918	0.800	0.801	0.785	0.792	0.800
492992	0.861	0.862	0.853	0.855	0.861
504607	0.717	0.726	0.709	0.712	0.726
624504	0.902	0.904	0.889	0.894	0.904
651739	0.814	0.809	0.804	0.808	0.806
651744	0.910	0.909	0.902	0.901	0.909
652065	0.832	0.832	0.805	0.803	0.828
average	0.820	**0.821**	0.808	0.812	0.819

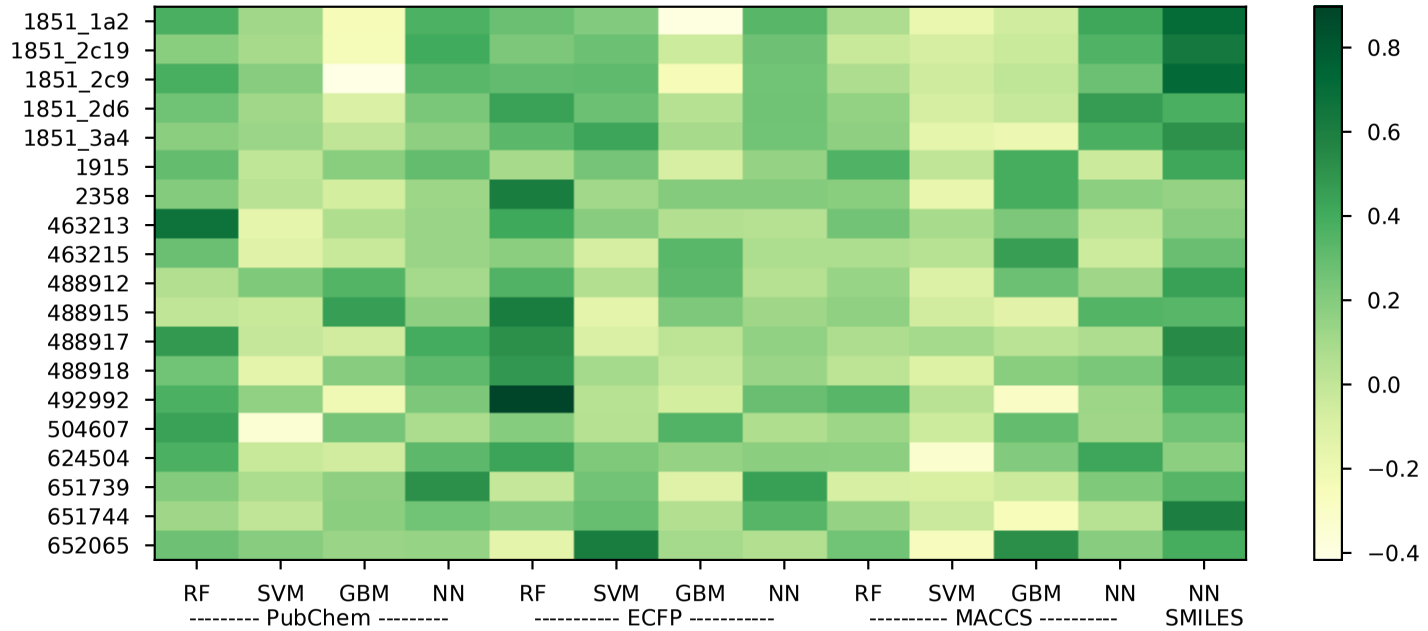

Figure 5.4: Model importance through the second-level learning. The absolute value of the coefficients through the second-level learning was used for model importance measurement. Darker green means highly weighted and significant model; lighter yellow means less weighted models.

formance of method ensemble; learning methods of RF, SVM, GBM, and NN were combined based on the same chemical representation. The method ensembles based on PubChem, ECFP, and MACCS showed 0.023, 0.021, and 0.024 higher performance than the average of each method classifier, respectively. The next five columns represent classification performance of representation ensemble; chemical representations of PubChem, ECFP, and MACCS were combined based on the same learning method. Unlike other machine learning problems, chemicals (drugs) have very diverse representation descriptors that express them. Ensemble can be performed on that basis. As with the method ensemble results, the ensemble between chemical representations outperformed the average of each representation classifier.

The comprehensive ensemble combines all the classifiers regardless of their learning method or representation; twelve classifiers combination of representations (PubChem, ECFP, and MACCS) and learning methods (RF, SVM, GBM, and NN), and SMILES-NN classifier. The comprehensive ensembles showed better performance than partial averaged ensembles, and comprehensive bayesian ensemble showed better performance than comprehensive average, and comprehensive meta-learning showed the best performance. For the meta-learning, we compared five learning algorithms. SVM showed the best meta-learning performance.

We analyzed the model importance through the learned weight. The *meta-learning* ensemble runs a second-level learning from predictions of the independent classifiers to make a final set of decisions. In this process, weights are assigned to each classifier and these weights can be interpreted as classifier importance. As represented in Figure 5.4, While PubChem-SVM and ECFP-GBM, and MACCS-SVM showed low importance, SMILES-NN and ECFP-RF showed high importance throughout the dataset. The SMILES-NN model did not show such high performance in a separate model, but it was regarded as the most important model in terms of the average coef-

ficient.

5.3.4 Ensemble Effects on Class Imbalance

Most bioassay datasets are class imbalanced; the imbalance ratio of the nineteen datasets used in our experiments was 1:2 in average. To see the ensemble effect on class imbalance, we compared plain neural network model (NN) with bootstrap aggregated neural network model (NN-bagging). The NN-bagging obtained from the averaged ensemble of 100 neural network models. Each neural network model was trained on N equal size data by sampling N samples from the original data uniformly and with replacement.

We experimented NN and NN-bagging on three types of representations (PubChem, ECFP, and MACCS) and analyzed the improved AUC on average. Figure 5.5a showed the improved AUC for each dataset; the improved AUC was calculated with AUC(NN-bagging)$-$AUC(NN). The performance of all datasets except `463215` was improved by using NN-bagging. We further analyzed the correlation between improved AUC and imbalance ratio, as represented in Figure 5.5b. The correlation coefficient between improved AUC and imbalance ratio was 0.69 with a p-value of 1.1×10^{-1}. In our experiments, we found that the more the class imbalance ratio was, the more improvement was achieved, through the ensemble of bagging.

5.4 Discussion

The fundamental of ensemble is considering all individual opinions and makes the final decision by combining the opinions. We tried to consider as many as possible diverse models and gave different weight on each model automatically, through learning the reliability of the opinions. We confirmed that comprehensive ensemble

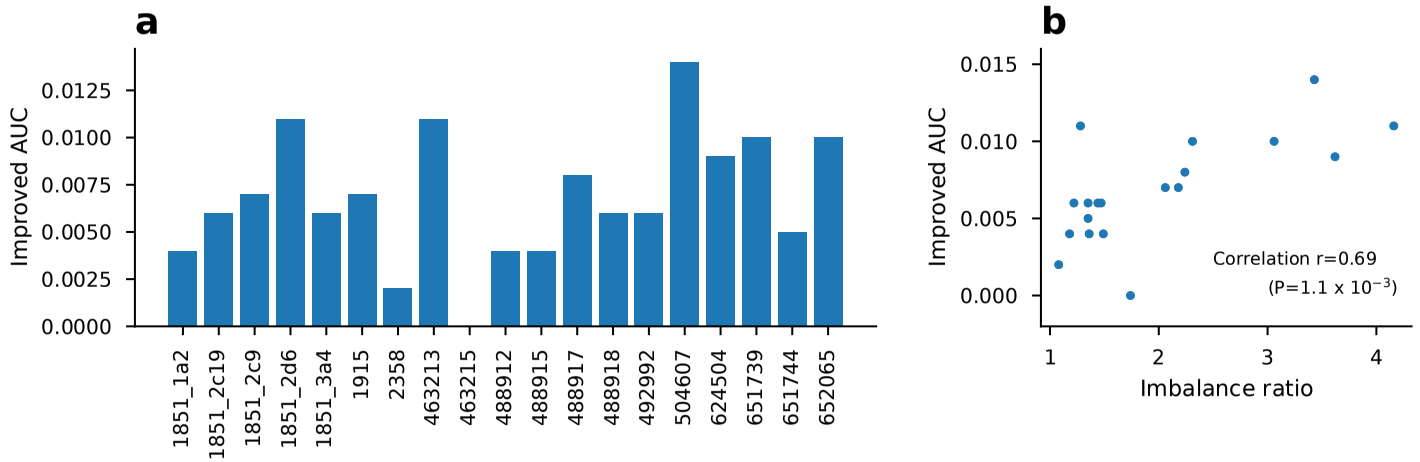

Figure 5.5: Ensemble effects on class-imbalanced datasets. **(a)** Improved AUC through bagging in Neural Network (NN) [AUC(NN-bagging)-AUC(NN)]. Each value is the averaged improved AUC over three fingerprints. **(b)** Pearson's correlation between the improved AUC and class imbalance ratio corresponding dataset (r=0.69, p-value=1.1x10^{-3}). The class imbalance ratio was calculated from the number of active and inactive chemicals, represented in Table 5.1.

showed better AUC than method ensemble, representation ensemble, and every other individual model. In combining technique, we confirmed that stacking showed better results than uniform averaging; four out of nineteen datasets showed better AUC in uniform averaging.

The proposed new type of individual model did not show such impressive results as the model alone. However, we could get hints from the second-level importance learning procedure that the model was a major predictor in combining with other methods. Only the proposed new single model takes a string type of SMILES as input. The other models use manually crafted vector type fingerprints. The string type of SMILES is not a proper format for the conventional methods, so we built a new neural network structure consisting of CNN and RNN to handle SMILES. Using this different approach, the proposed single method may capture important features hidden to the others.

Ensemble learning is simple but powerful. There is no reason not to use an ensemble if you want robust and trusted results.

Chapter 6

Conclusion

Sequential data is represented by a series of related things. Since a series of things in sequence data are not supposed to be i.i.d., it is inappropriate to use conventional machine learning methods without any modification. To learn hidden representations and their relatedness reside in sequential data, this dissertation introduces three methods besides k-mer: variable-order Markov model, 1D-CNN, and RNN.

As preliminary research, we used k-mer to find common patterns and exploited the k-mer frequency to determine the correct base when merging the paired-end reads. In the first study (Chapter 3), applying the variable-order Markov model and universal probability, we built a compact generalized model which avoids overfitting with the help of each technique. In the second study (Chapter 4), we automatically extracted the relevant features of two interacting chemicals with the help of a 1D-CNN with Siamese architecture. In the third research (Chapter 5), we applied a 1D-CNN and RNN on a string type chemical notation for QSAR prediction and found consistent importance exhibited by the model across diverse datasets.

The k-mer is the least complex approach utilized in the present research, and has been widely used for biological sequences and natural language. However, the

prediction model may be overfitted or underfitted depending on the k size.

The variable-order Markov model changes the k size flexibly depending on the sequence. As a result, it is more compact than the plain k-mer model, and is generalized. However, for an optimal compact model, it should calculate the probabilities of all possible context nodes.

Similar to k-mer, CNN filters of length l try to capture patterns across a sequence by sliding windows. The weight values (real number) of the filters are learned through back-propagation. A CNN can build a more complex model consisting of non-linear multi-layers with fewer parameters than plain k-mer.

Whereas CNNs focus on local dependencies, RNNs originally targets long term dependencies. The hidden value from the previous time linked to the current time and chained sequentially. RNNs (LSTM or GRU) generally requires more parameters than CNNs. The number of parameters and the sequential chain structure hinder the RNN from being calculated quickly in parallel.

This dissertation describes the features, advantages, and disadvantages of methods for sequential data. Based on related research and knowledge, we expect improved neural network architectures and probabilistic models that can memorize very long-distance clues and relatedness, as well as local dependencies.

Bibliography

[1] Dimitris K Agrafiotis, Walter Cedeno, and Victor S Lobanov. On the use of neural network ensembles in qsar and qspr. *Journal of chemical information and computer sciences*, 42(4):903–911, 2002.

[2] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[3] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215 (3):403–410, 1990.

[4] Ola Amayri and Nizar Bouguila. Improved online support vector machines spam filtering using string kernels. In *Iberoamerican Congress on Pattern Recognition*, pages 621–628. Springer, 2009.

[5] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7): 878, 2016.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural ma-

chine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Esben Jannik Bjerrum. Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*, 2017.

[8] Petko Bogdanov and Ambuj K Singh. Molecular function prediction using neighborhood features. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 7(2):208–217, 2010.

[9] Arthur Brady and Steven L Salzberg. Phymm and PhymmBL: Metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9):673–676, 2009.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Adrià Cereto-Massagué, María José Ojeda, Cristina Valls, Miquel Mulero, Santiago Garcia-Vallvé, and Gerard Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015.

[12] Lei Chen, Tao Huang, Jian Zhang, Ming-Yue Zheng, Kai-Yan Feng, Yu-Dong Cai, and Kuo-Chen Chou. Predicting drugs side effects based on chemical-chemical interactions and protein-chemical interactions. *BioMed research international*, 2013, 2013.

[13] Lei Chen, Jing Lu, Jian Zhang, Kai-Rui Feng, Ming-Yue Zheng, and Yu-Dong Cai. Predicting chemical toxicity effects based on chemical-chemical interactions. *PLoS One*, 8(2):e56517, 2013.

[14] Lei Chen, Jing Yang, Mingyue Zheng, Xiangyin Kong, Tao Huang, and Yu-Dong Cai. The use of chemical-chemical interaction and chemical structure

to identify new candidate chemicals related to lung cancer. *PloS one*, 10(6): e0128696, 2015.

[15] François Chollet. Keras: Deep Learning library for Theano and TensorFlow. `https://github.com/fchollet/keras`, 2015.

[16] Bertrand S. Clarke and Andrew R. Barron. Jeffreys' prior is asymptotically least favorable under entropy risk. *J. Statist. Plann. Inference*, 41(1):37–60, 1994.

[17] James R Cole, Qiong Wang, E Cardenas, J Fish, Benli Chai, Ryan J Farris, AS Kulam-Syed-Mohideen, Donna M McGarrell, T Marsh, George M Garrity, et al. The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic acids research*, 37(suppl 1):D141–D145, 2009.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, New York, second edition, 2006.

[19] Ryszard Czermiński, Abdelaziz Yasri, and David Hartsough. Use of support vector machine in pattern classification: Application to qsar studies. *Molecular Informatics*, 20(3):227–240, 2001.

[20] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[21] Arthur L Delcher, Douglas Harmon, Simon Kasif, Owen White, and Steven L Salzberg. Improved microbial gene identification with GLIMMER. *Nucleic Acids Research*, 27(23):4636–4641, 1999.

[22] Li Deng and Roberto Togneri. Deep dynamic models for learning hidden rep-

resentations of speech features. In *Speech and Audio Processing for Coding, Enhancement and Recognition*, pages 153–195. Springer, 2015.

[23] Todd Z DeSantis, Philip Hugenholtz, Neils Larsen, Mark Rojas, Eoin L Brodie, Keith Keller, Thomas Huber, Daniel Dalevi, Ping Hu, and Gary L Andersen. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Applied and environmental microbiology*, 72(7): 5069–5072, 2006.

[24] Jean-Pierre Doucet, Florent Barbault, Hairong Xia, Annick Panaye, and Botao Fan. Nonlinear svm approaches to qspr/qsar studies and drug design. *Current Computer-Aided Drug Design*, 3(4):263–289, 2007.

[25] Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280, 2002.

[26] Sean R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[27] Sean R Eddy. Accelerated profile HMM searches. *PLoS Comput Biol*, 7(10): e1002195, 2011.

[28] Robert C Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010.

[29] Jesse Eickholt and Jianlin Cheng. Dndisorder: predicting protein disorder using boosting and deep networks. *BMC bioinformatics*, 14(1):88, 2013.

[30] Andrea Franceschini, Damian Szklarczyk, Sune Frankild, Michael Kuhn, Milan Simonovic, Alexander Roth, Jianyi Lin, Pablo Minguez, Peer Bork, Chris-

tian Von Mering, et al. String v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(D1):D808–D815, 2013.

[31] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[32] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[33] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv preprint arXiv:1610.02415*, 2016.

[34] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[35] Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, Ajay Khanna, and Sean R Eddy. Rfam: an RNA family database. *Nucleic acids research*, 31 (1):439–441, 2003.

[36] Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean R Eddy, and Alex Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic acids research*, 33(suppl 1):D121–D124, 2005.

[37] Michael Gutman. Asymptotically optimal classification for multiple tests with

empirically observed statistics. *Information Theory, IEEE Transactions on*, 35 (2):401–408, 1989.

[38] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[40] Le-Le Hu, Chen Chen, Tao Huang, Yu-Dong Cai, and Kuo-Chen Chou. Predicting biological functions of compounds based on chemical-chemical interactions. *PLoS One*, 6(12):e29491, 2011.

[41] Xiaohua Hu. Using rough sets theory and database operations to construct a good ensemble of classifiers for data mining applications. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 233–240. IEEE, 2001.

[42] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[43] Hawoong Jeong, Sean P Mason, A-L Barabási, and Zoltan N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[44] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[45] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[46] W James Kent. BLAT—the BLAST-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[47] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[48] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[49] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[51] András Kocsor, Attila Kertész-Farkas, László Kaján, and Sándor Pongor. Application of compression-based distance measures to protein sequence classification: a methodological study. *Bioinformatics*, 22(4):407–412, 2006.

[52] Raphail E Krichevsky and Victor K Trofimov. The performance of universal encoding. *Information Theory, IEEE Transactions on*, 27(2):199–207, 1981.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[54] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238, 1995.

[55] Michael Kuhn, Christian von Mering, Monica Campillos, Lars Juhl Jensen, and Peer Bork. Stitch: interaction networks of chemicals and proteins. *Nucleic acids research*, 36(suppl 1):D684–D688, 2008.

[56] Greg Landrum. Rdkit: Open-source cheminformatics. *Online). http://www. rdkit. org. Accessed*, 3(04):2012, 2006.

[57] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[58] Byunghan Lee, Taehoon Lee, Byunggook Na, and Sungroh Yoon. Dna-level splice junction prediction using deep recurrent neural networks. *arXiv preprint arXiv:1512.05135*, 2015.

[59] Byunghan Lee, Junghwan Baek, Seunghyun Park, and Sungroh Yoon. deeptarget: end-to-end learning framework for microrna target prediction using deep recurrent neural networks. *arXiv preprint arXiv:1603.09123*, 2016.

[60] Taehoon Lee and Sungroh Yoon. Boosted categorical restricted boltzmann machine for computational prediction of splice junctions. In *ICML*, pages 2483–2492, 2015.

[61] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[62] Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

[63] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang. SOAP: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714, 2008.

[64] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22 (13):1658–1659, 2006.

[65] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[66] David Loewenstern, Haym Hirsh, Peter Yianilos, and Michiel Noordewier. *DNA sequence classification using compression-based induction*. Citeseer, 1995.

[67] Po-Ru Loh, Michael Baym, and Bonnie Berger. Compressive genomics. *Nature biotechnology*, 30(7):627–630, 2012.

[68] Jun Lu, Gad Getz, Eric A Miska, Ezequiel Alvarez-Saavedra, Justin Lamb, David Peck, Alejandro Sweet-Cordero, Benjamin L Ebert, Raymond H Mak, Adolfo A Ferrando, et al. MicroRNA expression profiles classify human cancers. *nature*, 435(7043):834–838, 2005.

[69] Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling*, 53 (7):1563, 2013.

[70] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in Bioinformatics*, page bbw068, 2016.

[71] HL Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[72] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[73] Chanin Nantasenamat, Chartchalerm Isarankura-Na-Ayudhya, Thanakorn Naenna, and Virapong Prachayasittikul. A practical overview of quantitative structure-activity relationship. 2009.

[74] Ka-Lok Ng, Jin-Shuei Ciou, and Chien-Hung Huang. Prediction of protein functions based on function–function correlation relations. *Computers in Biology and Medicine*, 40(3):300–305, 2010.

[75] Ragnar Nohre. *Some topics in descriptive complexity*. Ph.D. thesis, Linköping University, Linköping, Sweden, 1994.

[76] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[77] Rachid Ounit, Steve Wanamaker, Timothy J Close, and Stefano Lonardi. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC genomics*, 16(1):236, 2015.

[78] Hakime Öztürk, Elif Ozkirimli, and Arzucan Özgür. A comparative study of smiles-based compound similarity functions for drug-target interaction prediction. *BMC bioinformatics*, 17(1):128, 2016.

[79] Seunghyun Park, Seonwoo Min, Hyunsoo Choi, and Sungroh Yoon. deepmirgene: deep neural network based precursor microrna prediction. *arXiv preprint arXiv:1605.00017*, 2016.

[80] William R Pearson. [5] rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in enzymology*, 183:63–98, 1990.

[81] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[82] Christian Quast, Elmar Pruesse, Pelin Yilmaz, Jan Gerken, Timmy Schweer, Pablo Yarza, Jörg Peplies, and Frank Oliver Glöckner. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic acids research*, page gks1219, 2012.

[83] Christopher Quince, Anders Lanzén, Thomas P Curtis, Russell J Davenport, Neil Hall, Ian M Head, L Fiona Read, and William T Sloan. Accurate determination of microbial diversity from 454 pyrosequencing data. *Nature methods*, 6(9):639–641, 2009.

[84] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[85] Jorma Rissanen. Stochastic complexity and modeling. *Ann. Statist.*, pages 1080–1100, 1986.

[86] Gail Rosen, Elaine Garbarine, Diamantino Caseiro, Robi Polikar, and Bahrad Sokhansanj. Metagenome Fragment Classification using N-Mer Frequency Profiles. *Advances in bioinformatics*, 2008, 2008.

[87] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-tasser: a unified platform for automated protein structure and function prediction. *Nature protocols*, 5 (4):725–738, 2010.

[88] Jean-François Rual, Kavitha Venkatesan, Tong Hao, Tomoko Hirozane-Kishikawa, Amélie Dricot, Ning Li, Gabriel F Berriz, Francis D Gibbons, Matija Dreze, Nono Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062): 1173–1178, 2005.

[89] Steven L Salzberg, Arthur L Delcher, Simon Kasif, and Owen White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[90] Patrick D Schloss, Sarah L Westcott, Thomas Ryabin, Justine R Hall, Martin Hartmann, Emily B Hollister, Ryan A Lesniewski, Brian B Oakley, Donovan H Parks, Courtney J Robinson, et al. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and environmental microbiology*, 75(23): 7537–7541, 2009.

[91] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller.

Generating focussed molecule libraries for drug discovery with recurrent neural networks. *arXiv preprint arXiv:1701.01329*, 2017.

[92] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1):88, 2007.

[93] Xingjian Shi, Hao Wang, and Dit Yan Yeung. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. 2016.

[94] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[95] E Stackebrandt and BM Goebel. Taxonomic note: a place for DNA-DNA reassociation and 16S rRNA sequence analysis in the present species definition in bacteriology. *International Journal of Systematic Bacteriology*, 44(4):846–849, 1994.

[96] Andrew J Stuper, William E Brügger, and Peter C Jurs. *Computer assisted studies of chemical structure and biological function*. John Wiley & Sons, 1979.

[97] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[98] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[99] Matt Swain. PubChemPy: a way to interact with PubChem in Python. `http://pubchempy.readthedocs.io`, 2014.

[100] Mahmud Tareq Hassan Khan. Predictions of the admet properties of candidate drug molecules utilizing different qsar/qspr modelling approaches. *Current Drug Metabolism*, 11(4):285–295, 2010.

[101] Roman L Tatusov, Darren A Natale, Igor V Garkavtsev, Tatiana A Tatusova, Uma T Shankavaram, Bachoti S Rao, Boris Kiryutin, Michael Y Galperin, Natalie D Fedorova, and Eugene V Koonin. The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic acids research*, 29(1):22–28, 2001.

[102] Philippe Thomas, Mariana Neves, Illés Solt, Domonkos Tikk, and Ulf Leser. Relation extraction for drug-drug interactions using ensemble learning. *Training*, 4(2,402):21–425, 2011.

[103] Kai Tian, Mingyu Shao, Yang Wang, Jihong Guan, and Shuigeng Zhou. Boosting compound-protein interaction prediction by deep learning. *Methods*, 110: 64–72, 2016.

[104] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics, volume 41 (2 volume set)*, volume 41. John Wiley & Sons, 2009.

[105] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996.

[106] Han Van De Waterbeemd and Eric Gifford. Admet in silico modelling: towards prediction paradise? *Nature reviews Drug discovery*, 2(3):192–204, 2003.

[107] Marc HV van Regenmortel, Claude M Fauquet, David HL Bishop, EB Carstens, MK Estes, SM Lemon, J Maniloff, MA Mayo, DJ McGeoch, CR Pringle, et al. *Virus taxonomy: classification and nomenclature of viruses. Seventh report of the International Committee on Taxonomy of Viruses.* Academic Press, 2000.

[108] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[109] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[110] Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, and Stephen H Bryant. Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, 37(suppl 2):W623–W633, 2009.

[111] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. In *Proc. Edinburgh Math. SOC*, volume 17, pages 1–14, 1970.

[112] Travis J Wheeler and Sean R Eddy. nhmmer: DNA homology search with profile HMMs. *Bioinformatics*, page btt403, 2013.

[113] Jan Wildenhain, Michaela Spitzer, Sonam Dolma, Nick Jarvik, Rachel White, Marcia Roy, Emma Griffiths, David S Bellows, Gerard D Wright, and Mike Tyers. Systematic chemical-genetic and chemical-chemical interaction datasets for prediction of compound synergism. *Scientific Data*, 3, 2016.

[114] Frans M. J. Willems. The context-tree weighting method: extensions. *IEEE Trans. Inf. Theory*, IT-44(2):792–798, 1998.

[115] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. Context weighting: General finite context sources. In *14th Symp. Inf. Theory Benelux*, pages 120–127, Veldhoven, The Netherlands, May 1993.

[116] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol*, 15(3):R46, 2014.

[117] P Yap, J Kolberg, and M Urdea. Classification of hepatitis C virus into six major genotypes and a series of subtypes by phylogenetic analysis of the ns-5 region. *Journal of general virology*, 74:2391–2399, 1993.

[118] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[119] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

[120] Li Zhang, Haixin Ai, Wen Chen, Zimo Yin, Huan Hu, Junfeng Zhu, Jian Zhao, Qi Zhao, and Hongsheng Liu. Carcinopred-el: Novel models for predicting the carcinogenicity of chemicals using molecular fingerprints and ensemble learning methods. *Scientific reports*, 7(1):2118, 2017.

[121] Jacob Ziv. On classification with empirically observed statistics and universal data compression. *Information Theory, IEEE Transactions on*, 34(2):278–286, 1988.

[122] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530–536, 1978.

Abstract

시퀀스는 일련의 관계있는 것들의 나열로 정의 될 수 있으며, 일련의 단어로 구성된 자연어, 생물학적 시퀀스, 그리고 주식 시장이나 생물학적 신호의 시계열 데이터가 있다. 또한, 약물의 경우에도 화학적 성질을 나타내는 일련의 기호로 표현될 수 있다. 시퀀스에 존재하는 일련의 값들은 독립적이고 균일하게 분포되어있는 것이 아니라, 가변 길이에 근거리 혹은 원거리의 종속성을 가지고 있으며, 숨겨진 패턴을 내포한다.

이 논문은 순차적 데이터에 숨겨진 의존성과 패턴을 이해하는 세 가지 방법론을 제안한다. 첫째, 뉴클레오타이드 시퀀스의 정확하고 신속하며 확장 가능한 분류 요구에 따라, 가변 차수 마르코프 모델과 보편적 확률에 기반한 새로운 생성적 분류 방법을 제안한다. 다음으로 화학물간의 반응 (CCI)을 예측하기 위하여 수동 큐레이션 없이 화학적 특징을 자동으로 배우기 위해 Siamese 아키텍처를 기반으로 한 단대단 컨볼루션 신경망 구조를 제안한다. 모델은 단순화된 분자 입력 라인 입력 시스템 (SMILES)의 가공하지 않은 화학 기술자를 입력으로 사용한다. 마지막으로, 화학물의 구조와 반응 사이의 관계 (QSAR) 예측에 대한 견고한 결과를 얻기 위해 컨볼루션 및 재귀적 신경망 구조에 기반한 분류기와 앙상블 학습을 제안한다.

요약하면, 이 논문은 다양한 유형의 순차적 문제에 대한 방법론을 설명한다. 정보 이론에 기반한 접근법과 신경망 구조에 기반한 접근법이 정교하게 사용되었고, 견고한 성능을 확인하였다.

keywords: 기계학습, 심층학습, 신경망, 정보이론, 시퀀스분류, 약물분석

student number: 2014-31068