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

Large Scale Processing System for RNA Sequence Analysis

RNA 염기서열 분석을 위한 대규모 처리 시스템

2018년 2월

서울대학교 대학원

전기·컴퓨터공학부

김 한 주

Abstract

Since the early 2000s, the studies about ribonucleic acid (RNA) have come to a new turning point. As a result of the human genome project (HGP), only few percentage of genes are translated into proteins. From the other non-coding genes, many non-coding functional RNAs participate in various gene expression process. The researches of these functional RNAs, for example miRNA interaction, have advanced with the breakthrough in computer technology in the era of the big data.

In terms of volume and scale, the accumulated sequences and structures of RNA and their interactions can be treated from the viewpoint of big data. Thus, we expect the RNA analysis to be further improved by applying various techniques for large-scale processing on the scale of the data. In this thesis, we propose to building RNA interaction prediction database, structure comparison, and analysis. Additionally, we also propose a distributed framework for deep learning which is emerging as a new kind of approach for RNA analysis.

First, we build an integrated database for interactions between viral microRNA and host target genome. Many studies have reported on the interactions between autologous miRNAs and mRNAs for several species, however there has been no exploration of the interaction of viruses with their hosts. We predicted the interactions between miRNAs of the virus and the host mRNA using a variety of algorithms and constructed the computed results into a large-scale database, enabling rapid search and comparison. The second problem is about structures of RNA. We propose an online tool to quantify and analyze

the pairings on the RNA structure obtained through chemical methods instead of the physical method that requires high cost. In addition, we proposed a tree kernel based distance metric for RNA structures, and compared the tree-based distance measure methods that compare the known structure, and show that it assists classifiers in classifying the RNA structure by category. Finally, we propose the distributed deep learning framework for supporting RNA sequence comparison and identification. We propose an communication efficient algorithm and implementation for training deep neural networks with data parallelism in Apache Spark based commodity system, and show the speed-up by the results of image classification and sequence-to-sequence examples.

Keywords: RNA sequence, bioinformatics, large-scale data processing, database, machine learning, deep learning, parallel computing, distributed computing

Student Number: 2013-30232

Contents

List of Figures

List of Tables

Chapter 1

Introduction

RNAs have been shown to be related to a number of biochemical mechanisms, and many scientists and researchers have focused on RNAs and its interaction. Historically, RNAs have been considered that they just play a role of translation in the central dogma [20]. In detail, messenger RNA (mRNA), transfer RNA (tRNA), and ribosomal RNA (rRNA) have been known to participate in the protein translation. However, the discovery of non-coding functional RNAs such as microRNAs has opened up a new horizon for the RNA world. Although non-coding RNAs are not translated into protein directly, their genes comprise large proportion of the entire genes [25].

These non-coding functional RNAs perform various functions, such as suppressing or regulating gene expression, or splicing RNA molecules. Even in some viruses, RNAs are used as the genetic material itself. Similar to the proteins, these functional RNAs' structure implies the function of that RNA. Accordingly, we can approach to the true function of the functional RNAs closer by exploring its structure. Furthermore, we also infer the regulating function of an non-coding RNA from the interaction between the RNA and its

target whose function is well known. Consequently, the functions of non-coding RNAs are identified by exploring its structure or interaction with the other molecules. However, both of work are hard to achieve only by the traditional wet-lab experiments.

Among the non-coding functional RNAs, microRNAs (miRNAs) have drawn attention with the notable regulating mechanism [60, 85, 59, 57, 56]. miRNAs are small (19—24 nt in length) non-coding RNAs that down-regulate gene expression by binding to the 30 untranslated region ($3'$) of the target mRNA bearing complementary target sequences [5]. MiRNAs have been reported to control a wide range of biological processes such as hematopoiesis [13], neurogenesis [66], cell cycle control [29], and oncogenesis [42], indicating that miRNAs are core elements of the complete gene regulatory network, together with transcription factors. Because of a large number of combinations between miRNAs and its targets, predicting the interaction requires a lot of time and financial costs. Many computational methods have been suggested to screen the false positive RNA-RNA interactions [34, 38, 48, 62, 61, 73, 29, 28, 82, 94]. Some of these tools also provide its own large-scale interaction databases to store and search a large amount of predicting results easily, and the databases aid the wet-lab process in selecting target to be experimented.

This dissertation shows an example of building the integrated interaction database by proposing viral microRNA host target (vHoT) database in chapter 2. Some viruses have been reported to transcribe microRNAs, implying complex relationships between the host and the pathogen at the post-transcriptional level through microRNAs in virus-infected cells. Although many computational algorithms have been developed for microRNA target prediction, few have been designed exclusively to find cellular or viral mRNA

targets of viral microRNAs in a user-friendly manner. To address this, we introduce vHoT database for predicting interspecies interactions between viral microRNA and host genomes. Our database supports target prediction of viral microRNAs from human, mouse, rat, rhesus monkey, cow, and virus genomes. The contents of this chapter come from the following research:

- Hanjoo Kim, Seunghyun Park, Hyeyoung Min, and Sungroh Yoon, "vHoT: a database for predicting inter-species interactions between viral microRNA and host genomes," Archives of Virology, vol. 157, no. 3, pp. 497-501, March 2012.

Meanwhile, structural investigations for RNA molecules is also important for exploring the functional RNA world as well as predicting interactions. In chapter 3, we describe HiTRACE-web method theAlthough x-ray crystallography and nuclear magnetic resonance (NMR) spectrography have provided high resolution result, these approaches require high cost of experiments, and are limited by physical and chemical properties of RNAs [22]. Instead of the high-resolution approaches, the chemical methods was developed for probing RNA structure *in vivo* situation. In the chemical probing methods, capillary electrophoresis (CE) is one of the key techniques, and further provides a powerful and rapid means to map complex DNA and RNA structures at single-nucleotide resolution [101]. To facilitate the analysis of large-scale high throughput capillary electrophoresis data, we previously proposed a suite of efficient analysis software named HiTRACE (High Throughput Robust Analysis of Capillary Electrophoresis) [107]. HiTRACE has been used extensively for quantitating data from RNA and DNA structure mapping experiments, including mutate-and-map contact inference, chromatin footprinting, the Eterna RNA design project and other high-throughput applications. However, Hi-

TRACE is based on a suite of command-line MATLAB scripts that requires nontrivial efforts to learn, use and extend. Here, we present HiTRACE-Web, an online version of HiTRACE that includes standard features previously available in the command-line version and additional features such as automated band annotation and flexible adjustment of annotations, all via a user-friendly environment. By making use of parallelization, the on-line workflow is also faster than software implementations available to most users on their local computers.

In addition, a function of an RNA has a certain correlation with other biological molecules which has a similar structure to that RNA. In this regard, the comparative analysis is useful for comparing the structures and functions of RNAs with each other. The secondary structure of RNA can be represented in a tree form, and it is thus possible to calculate the (dis)similarity among RNAs in their tree representations. Using the pairwise (dis)similarity information, we can carry out further analyses, such as predicting the unknown function of RNA which has similar structure to a known RNA, or clustering RNA structures for unsupervised learning. In chapter 4, we calculate the (dis)similarity among test RNA structures sampled from real RNA data by using five different tree-based algorithms and a special type of the Mercer kernel, effectively bypassing difficulties in handling RNAs of different lengths and structures. We then use the k-nearest neighbor (k-NN) classifier to compare the performance of the tested algorithms in terms of classification accuracy. We also use the 5-fold cross validation method to compare the fidelity of predicting labels of the RNA test data for each algorithm.

The contents of chapter 3and4 come from the following researches:

- Hanjoo Kim, Pablo Cordero, Rhiju Das, and Sungroh Yoon, “HiTRACE-

Web: an online tool for robust analysis of high-throughput capillary electrophoresis," Nucleic Acids Research, vol. 41, no. W1, pp. W492-498, July 2013.

- Hanjoo Kim, Sungwoon Choi, Chulwoo Kim, and Sungroh Yoon, "Tree-based comparison and classification of RNA secondary structure," in Proceedings of the 28th International Technical Conference on Circuits/ Systems, Computers and Communications, Yeosu, Korea, July 2013.

Finally, we introduce deep learning platform which is capable of processing large-scale RNA sequence analysis. Deep models such as convolutional neural network (CNN) and recurrent neural network (RNN) have been successful in the biological sequence analysis [72]. Some previous studies have shown that RNN based models performs miRNA identification and its target prediction better than the alternatives [58, 80]. In the perspective of size and volume, RNA sequence and its structural information can be treated as big data already, and the amount of the data and computation continues to grow daily. Hence a distributed deep learning system that shares the training workload has been researched extensively.

However, combining deep neural network training with existing clusters often demands additional hardware and migration between different cluster frameworks or libraries, which is highly inefficient. Therefore, we propose a distributed deep learning framework which integrates the widely used data pipeline of Apache Spark with powerful deep learning libraries, Caffe and TensorFlow. Our framework embraces a brand-new parameter aggregation algorithm and parallel asynchronous parameter managing schemes to relieve communication discrepancies and overhead. It provides the fast and flexible

deep neural network training with high accessibility. We evaluated our proposed framework with CNN and RNN models; the framework reduces network traffic by 67% with faster convergence. The contents of chapter 5 come from the following researches:

- Hanjoo Kim, Jaehong Park, Jaehee Jang, and Sungroh Yoon, "DeepSpark: Spark-Based Deep Learning Supporting Asynchronous Updates and Caffe Compatibility," arXiv:1602.08191v2 [cs.LG], Februaray 2016.

To summarize, we provide descriptions of building a large scale database for miRNA-mRNA interactions in Chapter 2. Subsequently, we explain the detail about the structural exploration through chemical probing and computational approaches in Chapter 3, and the comparative analysis with existing structural information and functions in Chapter 4. In addition, we describe the distributed deep learning framework in Chapter 5.

Chapter 2

Building a database for RNA interactions

2.1 Background

Since the fundamental role of miRNA is gene regulation, the final goal of functional research on miRNA is often to find cognate targets of the miRNA, to elucidate the regulatory mechanism mediated by the miRNA, and to characterize its interactions with the target mRNA. To this end, many target prediction methods [34, 38, 48, 62, 61, 73] have been developed and have provided valuable tools for predicting the candidate targets of the miRNA of interest before wet-lab experiments are initiated for mechanism studies. However, those informatics tools are limited to predicting intra-species interactions between miRNA and mRNA targets in vertebrates and some lower animals such as flies and worms. Little has been done to predict interspecies interactions and especially to find the interactions between host targets and miRNAs of parasites such as viruses.

A number of viruses (mostly DNA viruses) have been shown to encode miRNAs [29, 28, 82, 94], and it is intriguing to see the effects of viral miRNAs on viral pathogenesis. Viral miRNAs have some distinct features compared to conventional miRNAs. Although most animal miRNAs down-regulate gene expression through incomplete binding to target sequences that is not accompanied by target cleavage, some virus-encoded miRNAs such as miR-BART2 from Epstein-Barr virus (EBV) have been shown to act in an siRNA-like manner [82]. In addition, in contrast to common cellular miRNA-mRNA interactions, there exist complex relationships between miRNAs and mRNA targets in virus-infected host cells, because infected cells possess two different (i.e., host and viral) genomes. Cellular miRNAs may affect the expression of viral mRNA targets as well as cellular targets, and similarly, viral miRNAs may inhibit the expression of host mRNAs and virus mRNAs. In other words, two additional interspecies interactions, namely the interaction between viral miRNA and host mRNA, and the interaction between host miRNA and viral mRNA, are present in virus-infected host cells. These interspecies interactions are critical for understanding the life cycle of a virus and its pathogenesis. Additionally, given that there are examples of viral miRNAs whose target regions are present not in the $3'$ UTR but in the coding region [29], the $5'$ UTR and coding sequences as well as the $3'$ UTR need to be considered when putative mRNA targets are computationally predicted.

Although many computational algorithms have been developed to predict candidate miRNA targets, few have been designed exclusively to find cellular or viral mRNA targets of viral miRNAs. Recently, two computational methods, Reptar [55] and miRiam [26], have been developed to predict host mRNA targets of viral miRNAs, but there are limitations to using those tools in a vi-

ral miRNA study. miRiam is not based on a user-friendly web application but instead requires the Python interpreter to run the program, and thus, users who are unfamiliar with Python may not have easy access to this method. RepTar has been reported to predict putative cellular targets of viral miRNAs, but queries are limited to viral miRNAs derived from Epstein-Barr virus (EBV), human cytomegalovirus (HCMV), Kaposi's sarcoma virus (KSHV), mouse cytomegalovirus (MCMV), and mouse gammaherpesvirus (MGHV).

2.2 Method

To address these limitations of the existing approaches, we introduce the viral miRNA host target (vHoT) database, a web-based tool that can search for mRNA targets of viral miRNAs. The vHoT database allows interspecies analysis and can thus be used to predict both cellular mRNA targets and viral mRNA targets of virus-derived miRNAs. The current version of vHoT can predict the targets of 271 viral miRNAs within the human, mouse, rat, rhesus monkey, cow, and virus genomes. Either a user-specified set of genes or the whole genome of an organism can be used for prediction. Five widely used algorithms that are known to be relatively accurate for target prediction, TargetScan [62, 61], miRanda [43, 7], RNAhybrid [84], DIANA-microT [48, 68] and PITA [45], were customized and used as the search engines of vHoT, and the user can specify the key parameters of these algorithms to fine-tune search results. Several aspects of the internal database and the user interface of vHoT were optimized to maximize the operational efficiency as well as the quality of the user experience.

Figure 2.1 shows the overall architecture of the proposed vHoT database.

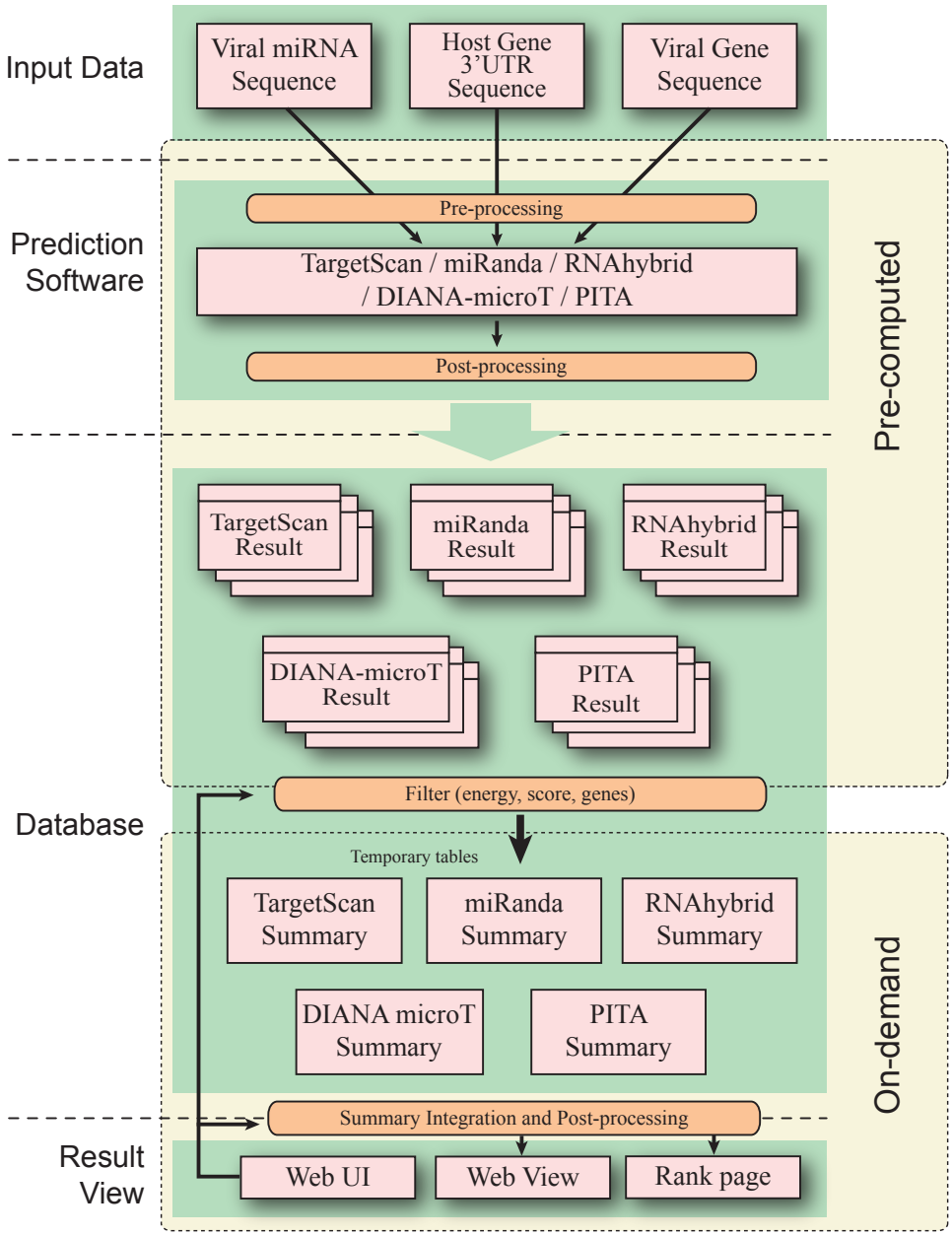

Figure 2.1: Overall system architecture of the proposed vHoT database. There are two phases depicted in the figure: the pre-computation and the on-demand phases. The pre-computation phase occurs during the initial development of the database or whenever updates for prediction algorithms become available. The activities in the on-demand phase occur when the database is up and running

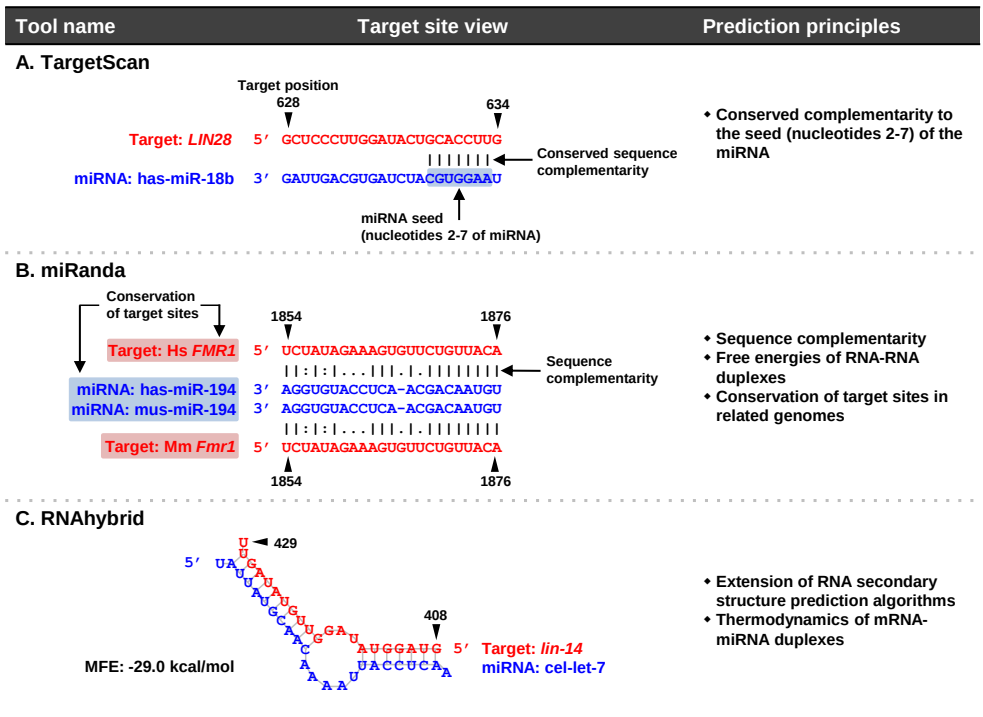

Figure 2.2: Target-prediction algorithms and their prediction principles. **(A)** TargetScan predicts miRNA targets by searching for conserved sites that match the miRNA seed region (nucleotides 2–7). Shown are miRNA has-miR-18b and its LIN28 target. **(B)** miRanda relies on three key prediction principles, namely sequence complementarity, the free energy of an RNA-RNA duplex and conservation of target sites in related genomes. Shown are conserved miRNAs has-miR-194 and mus-miR-194 and their targets Hs FMR1 and Mm Fmr1. **(C)** RNAhybrid predicts miRNA targets by finding the minimum free energy (MFE) of hybridization of miRNA-mRNA duplexes. Shown are miRNA cel-let-7 and its target lin-14, which form a duplex with an MFE of -29.0 kcal/mol

It takes three types of sequences as input: viral miRNA sequences; human, mouse, rat, rhesus monkey, and cow 3′ UTR sequences; and viral gene sequences. For the current release of vHoT, we obtained a total of 271 viral miRNA sequences from release 15 of miRBase [30, 31, 32, 33]. For human and mouse, we downloaded approximately 89,530 mRNA 3′ UTR sequences from the TargetScan web site (release 5.1). We also utilized 20 viral genes available in the 2006 release of the ViTa database[38]. As the prediction engine of vHoT, we customized five widely used miRNA target prediction tools: TargetScan, miRanda, RNAhybrid, DIANAmicroT and PITA. When making a decision on which tools to use, we considered various aspects of the tool, such as its quality of prediction, how well it is maintained, the ease of data extraction and customization, and its popularity and acceptance in the community.

The target prediction algorithms and their prediction principles are depicted in Figure 2.2. TargetScan is software to predict miRNA targets by searching for conserved sites that match the miRNA seed region. As of June 2011, the TargetScan database lists 3,088 miRNAs and 410,320 mRNA 3′ UTR sequences for 10 species. miRanda utilizes three key principles, namely sequence complementarity, the free energy of an RNA-RNA duplex, and conservation of target sites in related genomes. As of October 2011, miRanda provides 851 human, 793 mouse and 698 rat miRNAs and many more miRNAs derived from 19 species other than viruses. RNAhybrid predicts miRNA targets by finding the minimum free energy (MFE) of hybridization of miRNA-mRNA duplexes. RNAhybrid was originally developed as an extension of an RNA secondary structure prediction algorithm called BiBiServ RNA Studio [87]. DIANA-microT uses a 38-nt window for progressively scanning 3′ UTRs of candidate targets and calculates the minimum binding energy of

miRNA-target duplexes by dynamic programming. DIANA-microT has been reported to have high precision levels greater than 66% [88]. PITA employs a parameter-free interaction model that computes the difference between the free energy of a miRNA-target duplex and the energetic cost of unpairing the target for making it accessible to the miRNA. PITA demonstrates that target accessibility is key in miRNA function.

2.3 Implementation

For implementation, there are two phases in vHoT, as depicted in Figure 1: the pre-computation and on-demand phases. The pre-computation phase occurs during the initial development of the database, or whenever updates for prediction algorithms become available. We first appropriately preprocess the three types of input sequences mentioned above so that the five prediction tools used can take them as input. Using the preprocessed sequence information, we then execute the five miRNA target prediction tools. To provide the user of vHoT with near-real-time response we compute a vast number of search results for each prediction software in advance and insert them into the MySQL database management system [6]. For efficient insertion, we can use the low-level IO functionality of MySQL and/or the database interface (DBI) module of the PERL language [11]. As a database engine, we employ the MyISAM storage engine, which is optimized for read-dominant applications such as vHoT, over the InnoDB engine [110].

The activities in the on-demand phase occur when the database is up and running. The user can issue a query after customizing it with various search options and conditions that vHoT supports. For instance, the user can adjust

the parameters of individual prediction tools and decide whether to use the union or intersection operation to combine the results from individual tools.

According to the user-specified filtering conditions and parameters, vHoT then creates a set of intermediate tables from the pre-computed results and processes these tables to generate the search result. To highlight the most important findings for the user, vHoT supports various options to rank the search result. Further details on vHoT can be found on the website (`http://dsl.snu.ac.kr/vhot/`).

The identification of miRNAs from viruses brought in a new layer of gene regulation affecting virus-host interactions. Many reports have shown that viruses use miRNAs to regulate their life cycles and evade host immune surveillance [86], but studies on viral miRNAs still lag behind those on human miRNAs or mouse miRNAs. With the development of vHoT, we expect that researchers investigating viral miRNAs will find it much easier to search for putative cellular and viral targets of their viral miRNAs of interest. We further anticipate that vHoT will contribute to elucidating the mechanism of viral pathogenesis by revealing interactions between miRNAs and cellular mRNAs. One limitation of the current version of vHoT is the number of species supported: vHoT considers humans, mice, rats, rhesus monkeys, cows and viruses as target species. The next release of vHoT will include more species, enabling the analysis of host-specific interspecies interactions of viruses with more types of host genomes.

For user convenience, it would also be possible to highlight among the search results the experimentally validated targets of viral miRNAs by utilizing the information available in the TarBase 5.0 [89] and miRecords [106] databases.

Chapter 3

Structural analysis pipeline for RNA structures

3.1 Background

Capillary electrophoresis (CE) is one of the most powerful and widely used nucleic acid separation techniques available [102]. Its conventional application areas include genomic mapping, forensic identification, and genome sequencing [67, 105]. Recently combined with chemical probing methodologies, CE further provides a powerful and rapid means to map complex DNA and RNA structures at single-nucleotide resolution [101].

In chemical-probing-based RNA structure inference studies, a chemical reagent modifies the RNA of interest, either cleaving it or forming a covalent adduct with it. Commonly used reagents include hydroxyl radicals [22, 46], dimethyl sulfate alkylation (DMS) [93], carbodiimide modification (CMCT) [97], and the SHAPE strategy using 2′-OH acylation [71]. Subsequent reverse transcription detects the modification sites as stops to primer extension at nu-

cleotide resolution. Traditionally, the resulting cDNA fragments were resolved in sequencing gels followed by individually quantifying band intensities. To resolve these fragments in a high-throughput fashion, capillary electrophoresis (CE) can be used. CE-based chemical probing can produce tens of thousands of individual electrophoretic bands from a single experiment, leading to recent breakthroughs in high-throughput nucleic acid structure mapping: *e.g.,* automated inference of complex RNA structures [101, 74, 95, 21, 50] such as ribosomes [24], and viruses [104, 100].

Analyzing a large number of electrophoretic traces from a high-throughput structure-mapping experiment is time-consuming and poses a significant informatics challenge. It requires a set of robust signal-processing algorithms for accurate quantification of the structural information embedded in the noisy traces. Current software methods for CE analysis include capillary automated footprinting analysis (CAFA) [74], ShapeFinder [95], high-throughput robust analysis for capillary electrophoresis (HiTRACE) [107], fast analysis of SHAPE traces (FAST) [79], and QuShape [44]. The most recent of these methods have largely converged on the basic steps of analysis: preprocessing (such as selection of the data-containing range and baseline adjustment), deconvolution of co-loaded mapping signal traces and reference traces, alignment, peak detection, band annotation, peak fitting, signal decay and background subtraction.

Although these programs are all useful for semi-automated CE data analysis, they suffer from certain limitations. CAFA has effective peak fitting capabilities but lacks alignment and annotation features, thus necessitating laborious efforts for analyzing multiple capillaries. ShapeFinder and QuShape provide sophisticated signal alignment, annotation, and peak fitting capabilities, but have limited cross-capillary analysis capabilities, making these tools less

efficient for analyzing data from large-scale experiments involving hundreds of capillaries. FAST is highly optimized for rapid band annotation but relies on the proprietary ABI utility programs, which has made it difficult to customize and extend to various experimental scenarios. HiTRACE is feature-rich and has been extensively used for high-throughput structure mapping studies such as the mutate-and-map strategy [50, 49, 51], chromatin footprinting, and the massively parallel RNA design project EteRNA [8]. However, HiTRACE is a suite of command-line MATLAB scripts that requires nontrivial efforts to learn, use, and extend, and a purchased license (The MathWorks, `http://www.mathworks.com`).

HiTRACE-Web is based on the HiTRACE workflow and thus inherits all of its core functionalities for high-throughput CE analysis. Going one step further, HiTRACE-Web provides an integrated and interactive on-line interface. In addition to the standard features previously available, HiTRACE-Web presents additional features such as automated band annotation and adjustment. By making use of powerful multi-core server processors, HiTRACE-Web also runs faster than the previous off-line implementations available to most users on their local computers. To the best of the authors' knowledge, HiTRACE-Web is the first on-line tool for high-throughput CE data analysis.

3.2 Method overview

HiTRACE-Web takes as input a set of nucleic acid structure mapping profiles obtained from a number of capillaries. A profile represents the intensity of a nucleic acid sample in a capillary as a function of electrophoretic time. The peaks in a profile appear as bands in a gray-scale image. Band/peak locations

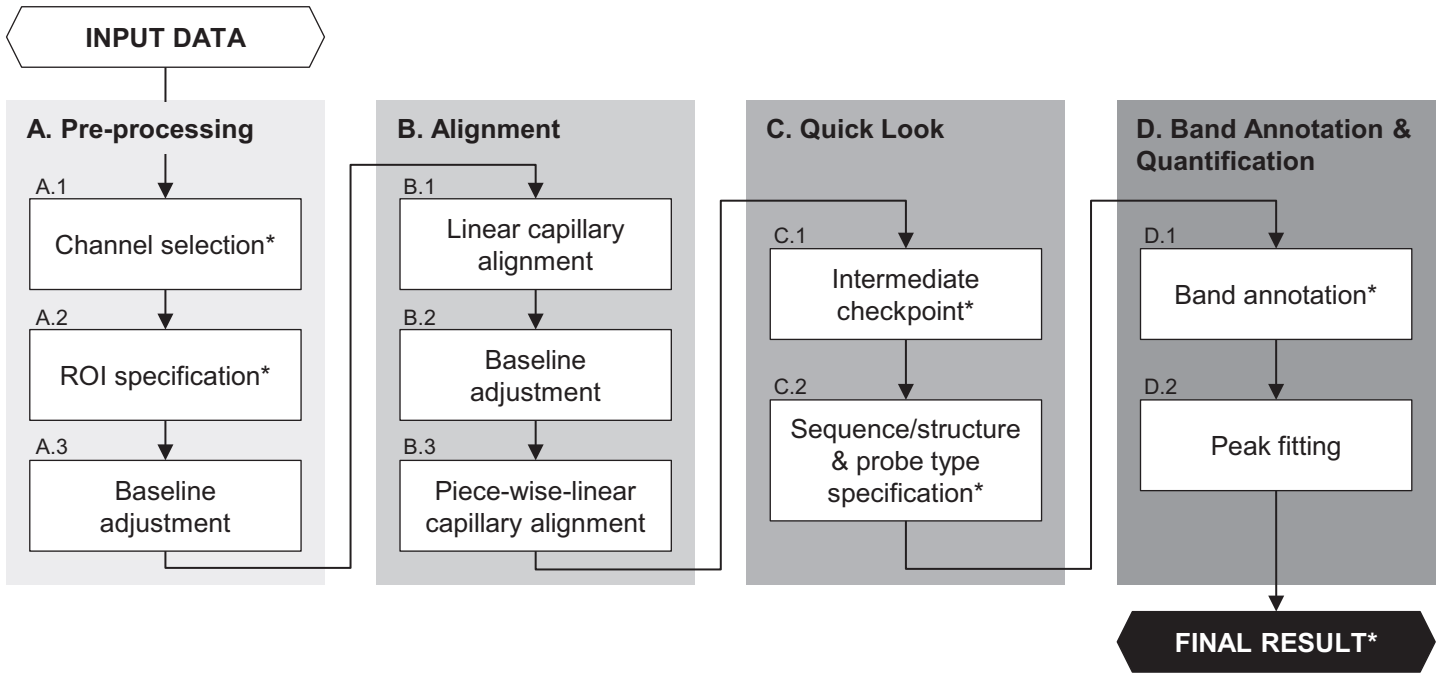

Figure 3.1: Flowchart of HiTRACE-Web analysis pipeline.

represent individual residues of a nucleic acid sequence. Commonly used ABI sequencers (Life Technologies Corporation, Carlsbad, USA) can separate the products of hundreds of nucleotides, and we assume that each CE profile contains hundreds of bands. The final output of HiTRACE-Web is a set of aligned and annotated profiles with quantified band areas in text and image formats. To deliver the output, HiTRACE-Web works in multiple stages interleaved with user checkpoints: preprocessing, profile alignment, band annotation, peak fitting and quantification.

Figure 3.1 shows the flowchart of the analysis pipeline. Each of the steps marked with an asterisk corresponds to a tab in the web server implementation. The first stage is preprocessing in which the user can confirm or refine the signal and the reference channels, proper data regions for analysis, and the correction of constant baselines. HiTRACE-Web carries out profile alignment in the second stage. Both linear (for within-batch and between-batch alignment) and piece-wise-linear alignment is performed. After alignment, the user can check the intermediate result, provide sequence and structure information, and specify types of chemical reagents in the third stage. In the next stage, HiTRACE-Web provides an automated band annotation functionality, which allows users to complete initial assignments of hundreds of bands in hundreds of profiles in the order of seconds. HiTRACE-Web then performs peak fitting to approximate a profile as a sum of Gaussian curves.

In preprocessing, the user needs to provide information on capillary channel compositions. To facilitate the analysis process, it is common in typical CE experiments to co-load samples with a reference ladder which fluoresces in a different color. Multiple spectral channels thus exist in each capillary, and the user should specify which channels contain the structure mapping signal and

the reference (typically a ladder) (Step A.1).

The next step in preprocessing is to select the range of analysis, since typical profiles carry information only within a subset of the entire electrophoresis run. HiTRACE-Web utilizes edge-detection techniques [12] to select proper regions for further analysis automatically (Step A.2). It is also possible for the user to set the region manually. HiTRACE-Web then carries out a series of additional preprocessing operations on the selected region of capillaries such as adjustment of constant baseline (Step A.3). More details of the preprocessing operations can be found in [107]. The second stage in HiTRACE-Web is alignment. Profiles need to be aligned to each other because the products in different capillaries are subject to slightly different electrophoretic conditions and their detection times are then shifted and scaled compared to each other. Due to the finite number of capillaries an ABI sequencer can handle at a time (*e.g.,* 96 for ABI3730), CE experiments using hundreds of capillaries consist of multiple batches. We designate the first capillary in each batch as the reference and use it for within-batch alignment. There, we align each profile to the reference by finding the optimal shift and scaling amounts that maximize its correlation to the reference (part of Step B.1). After the within-batch alignment, we carry out additional alignment procedures: between-batch adjustment for global alignment (part of Step B.1), adjustment of smooth baseline (Step B.2), and dynamic-programming-based piece-wise-linear alignment for fine-tuning local disagreements [107] (Step B.3).

After alignment, the user checks intermediate results (Step C.1) and specifies the sequence probed and the type of chemical modifications applied to each capillary (Step C.2). The user can currently select among nine options: three types of chemical reagents (DMS, CMCT, and SHAPE), four types of

dideoxynucleotides (ddGTP, ddATP, ddTTP, and ddCTP), no modification, or other. HiTRACE-Web provides a convenient and intuitive graphical interface for selecting signal and reference channels and specifying capillary modification types. This specification provides HiTRACE-Web with information on where bands should appear in the data, *e.g.,* primarily at A and C positions for DMS modification of RNA data [18], and with annotations carried forward to the final result files (see below). Following the profile alignment is the band annotation procedure (Step D.1), which refers to the process of mapping each band in an electrophoretic profile to a position in the nucleic acid sequence. For verification, visual inspection of band annotation results is normally inevitable to certain extent, but the manual band annotation step takes significant human efforts when there are a large number of profiles. To address this issue, HiTRACE-Web provides an automated band annotation functionality, which allows users to complete initial assignments of hundreds of bands in hundreds of profiles in the order of seconds. More details of the automated band assignment algorithm are beyond the scope of this server-focused chapter and will be described elsewhere. HiTRACE-Web also provides an interactive interface to adjust the band annotation manually so that users can correct any suboptimal assignment they find. In the last stage, HiTRACE-Web performs peak fitting (Step D.2) to approximate a profile as a sum of Gaussian curves, each of which is centered at the intensity peak location. The peak amplitudes are selected in the least-squares sense by using a standard optimization technique, thus minimizing the deviations of the Gaussian model from the CE profiles [107]. As the final output, HiTRACE-Web reports the peak quantification results including the area and location of each band and exports data annotations, sequence, and structure to an

RNA data (RDAT) formatted file [19]. The RDAT file can then be submitted to the RNA Mapping Database (RMDB) repository for data sharing (`http://rmdb.stanford.edu/submit`) or structure server for secondary structure model estimation (`http://rmdb.stanford.edu/structureserver`) [19], and links to these tools are provided.

HiTRACE-web leaves correction for attenuation of band intensity for longer products ('signal decay'), background subtraction, and final normalization to the user. These post-processing tasks are carried out differently by independent groups who have made distinct *ad hoc* assumptions [44, 52, 3], and indeed these tasks are left out of some applications, such as the mutate-and-map technique [51], due to the introduction of noise. Robust experimental and computational approaches for post-processing chemical mapping data are in development (T. Mann, PC, RD, in prep.), as are additional features including secondary structure display and error estimation in automatic sequence assignment. Inclusion into HiTRACE-Web will allow these advances to be disseminated rapidly to the RNA structure mapping community.

3.3 Web server

The HiTRACE-Web server utilizes the Apache HTTP Server (The Apache Software Foundation, `http://httpd.apache.org/`) for basic web services and the MySQL Server (Oracle Corporation, `http://www.mysql.com/`) for internal data management. For client-side programming, we used CodeIgniter (EllisLab, `http://ellislab.com/codeigniter`), an open-source web application framework, and coded the web pages in PHP (The PhP Group, `http://php.net/`) and JavaScript (Mozilla Foundation, `https://developer.mozilla.`

Figure 3.2: Overview of data analysis using HiTRACE-Web.

org/en/docs/JavaScript) with jQuery (The jQuery Foundation, http://jquery.com/) and jQuery user-interface (UI) plugins. Interactive UI components also use the canvas elements defined in the HTML5 standard (World Wide Web Consortium, http://www.w3.org/TR/html5/). We used Ajax (http://en.wikipedia.org/wiki/Ajax_(programming)) for asynchronous data transfers between the client and server sides. On the server side, we used Gearman (http://gearman.org), an open-source application framework, to schedule multiple requests. Each user request is handled by a worker written in PHP. The worker creates template code in MATLAB that contains input parameters and links to the user data. The worker also forks the MATLAB interpreter so that it can execute the template code. Most of the time-consuming operations are multi-threaded, and the current HiTRACE-Web server runs on a 48-core machine (four on-board AMD Opteron 6172 processors with 256GB main memory; Ubuntu Linux version 3.2.0-29). In this environment, processing 52 capillaries (with 60 bands per capillary) takes a few minutes including manual adjustments. HiTRACE-Web supports most of the widely used web browsers including Google Chrome (Version 25.0 or later; http://www.google.com/chrome/), Mozilla Firefox (Version 19.0 or later; http://www.mozilla.org/), Apple Safari (Version 6.0 or later; http://www.apple.com/safari/), and Microsoft Internet Explorer (Version 9.0 or later; http://windows.microsoft.com).

Figure 3.2 [1] explains the overall flow of CE data analysis using HiTRACE-

[1](A) Users can select the reference and signal channels using the graphical interface. (B) HiTRACE-Web provides a feature to select the valid region of interest automatically. Alternatively, users can set the range manually. The red rectangles in the figure represent the regions of interest. (C) Before advancing to the next step, HiTRACE-Web provides a snapshot of intermediate results, so that the user can go over the previous steps if needed. (D) HiTRACE-Web provides an intuitive graphical interface to specify the chemical modifications made to capillaries. For each of them, the user can select among nine color-coded options: three chemical reagents (dimethyl sulfate alkylation (DMS) [93], carbodiimide modification

Web. The input file is a zip-compressed collection of .ab1 or .fsa files from ABI sequencers in the ABIF file format. If there are multiple batches, each batch should be presented in a subfolder, as is the typical output from ABI sequencers. After uploading the input file, the user needs to specify the signal and reference channels. HiTRACE-Web provides a graphical interface for channel selection (Figure 3.2A). Next, the region of interest is specified either manually or automatically (Figure 3.2B). HiTRACE-Web then carries out the alignment of the profiles in the selected region, following the procedures outlined previously. The user can visually inspect the intermediate result after alignment (Figure 3.2C) and go over the previous stages if needed. If satisfactory, the user can start specifying the type of chemical modification data present in each capillary using the graphical interface HiTRACE-Web provides (Figure 3.2D). The user can then perform band annotation (Figure 3.2E). HiTRACE-Web permits either manual or automated band annotation; in practice, both types of annotations complement each other: performing automated band annotation first and then adjusting the result manually allows a large number of bands to be annotated quickly and robustly. After band annotation, HiTRACE-Web carries out peak fitting and quantification. The results are provided as downloadable images (Figure 3.2F), tab-delimited text files (with quantified areas; one column per capillary and one row per residue), and an RDAT file that

(CMCT) [97], the SHAPE strategy using 2′-OH acylation [71]), reference ladders using four dideoxynucleotides (ddGTP, ddATP, ddTTP, and ddCTP), no modification, and other. (E) HiTRACE-Web can carry out band annotation in an automated fashion, thus reducing the analysis time substantially. Furthermore, HiTRACE-Web provides a user-friendly interface for users to fine-tune band annotation results manually. Red circles represent the residue locations. Each type of nucleotide is associated with a different color (G: green, C: cyan, U: blue, and A: red). By clicking the image, user can (de)select the position of individual residues. The auto-assigned bands are shown on the right side in gray for easier referencing when manually adjusting the assignment. (F) HiTRACE-Web reports a set of aligned and annotated profiles with quantified peak areas in the image, tab-delimited text, and RDAT (25) formats users can download for further uses.

can be submitted to the repository and structure modeling server available at the RNA Mapping Database [19]. More detailed instructions to each analysis stage and explanations of results are available at the HiTRACE-Web homepage. A complete tutorial with example data is also available, with specific help at each step.

Figure 3.3A and 3.3B show the correlation of quantification results between HiTRACE-Web and HiTRACE [107] for two different users. The high level of Pearson's correlation coefficients (R^2 of 0.9996 and 0.9999; $P < 0.001$) between band intensities quantified with HiTRACE-Web and those quantified with HiTRACE indicates lack of any major systematic deviations induced by HiTRACE-Web. To test the consistency in band quantification between different users, we let two independent users carry out quantification of the same data using HiTRACE-Web and HiTRACE, as shown in Figure 3.3C and 3.3D, respectively. Both tools resulted in excellent agreement between the independent analyses (R^2 of 0.9993 and 0.9986; $P < 0.001$). The data used is from an RNA structure mapping study using the mutate-and-map strategy [50, 49, 51]. The mapped sequence length was 92 nucleotides, and the data describes six different mapping experiments including SHAPE, DMS and reference ladders. The total number of bands was 552.

3.4 Summary

We developed HiTRACE-Web (freely available at `http://hitrace.org`), an online server for rapid and robust analysis of large collections of profiles obtained from high-throughput CE experiments. Recent generations of high-throughput DNA and RNA structure mapping studies give hundreds of CE

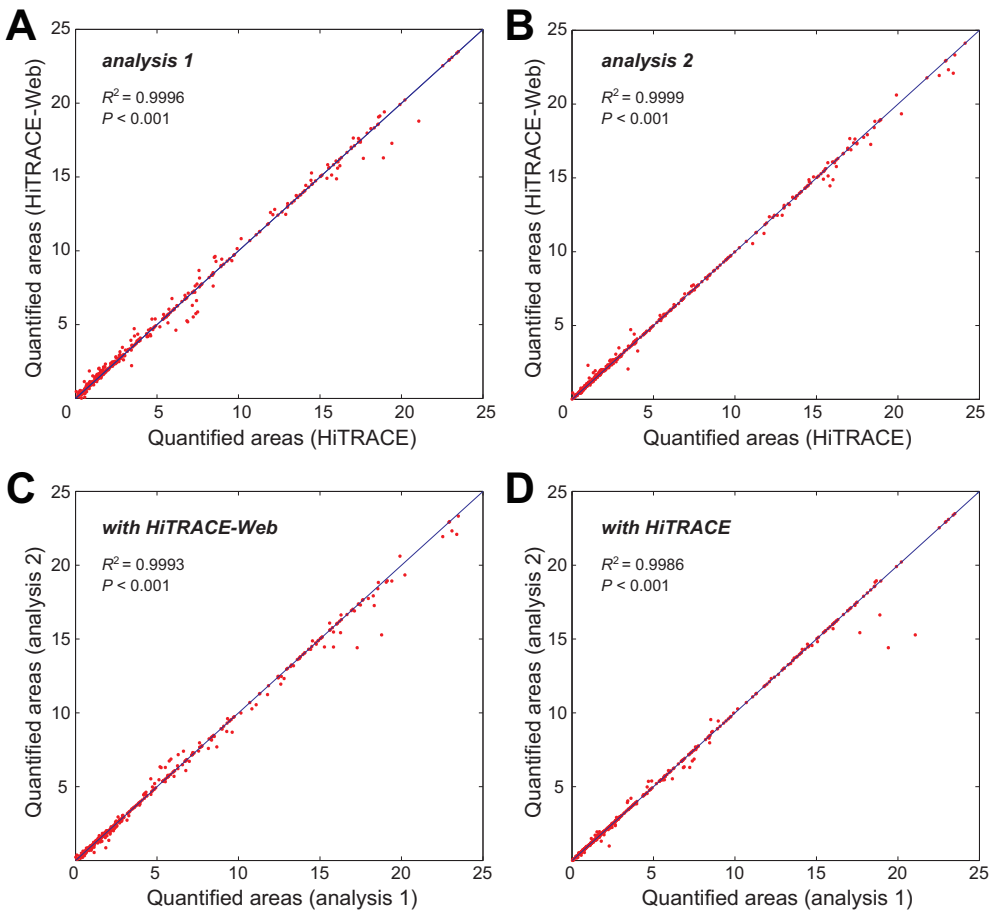

Figure 3.3: Confirming consistency in band quantification results between tools and analyses. (A-B) Correlation of quantification results between HiTRACE-Web and HiTRACE [107] for two independent users. These plots indicate lack of any major systematic deviations induced by HiTRACE-Web. (C-D) HiTRACE-Web gives the same level of consistency between independent analyses as HiTRACE. The data used is from an RNA structure mapping study using the mutate-and-map strategy [50, 49, 51]. A 92-nucleotide RNA sequence was treated in six different mapping conditions including SHAPE, DMS and ddTTP, giving 552 bands in total.

profiles each containing hundreds of bands. To isolate signals from such a large collection of profiles, we previously developed a signal-processing pipeline that consists of multiple stages including preprocessing, alignment, annotation, and band quantification. HiTRACE-Web now enables users to follow the whole pipeline through a user-friendly, integrative, and interactive web environment. It is our hope that HiTRACE-Web can contribute to large-scale structure inference studies based on chemical probing and CE separation by providing an effective and easily accessible data analysis framework.

Chapter 4

RNA structural representation

4.1 Background

The structure information of a non-coding, functional RNA has a direct relationship to its function [69]. Methods that compare different RNA structures can thus be a pathfinder for identifying unknown RNA functions from structure. In this chapter, we test five different RNA structure comparison algorithms to measure the (dis)similarity among RNA structures. These target comparison algorithms are based on trees and use structural edit-distance metrics. In the method section, we introduce basic algorithms for computing edit distance between two trees. Each RNA secondary structure is represented by the dot-bracket notation (DBN) [37, 113, 90, 91, 27]. A dot represents an unpaired residue, and a pair of brackets stands for base pairing. In addition, a DBN form can be translated into a tree form [37]. Figure 4.1 shows an example of a secondary structure represented in DBN and its corresponding tree

structure. For an RNA structure, each node in a tree is associated with a value indicating whether the node represents base pairing or not. For trees, an edit distance metric can be used to measure (dis)similarity between two different RNA structures. And we applied kernel-based tree distance metric which is used in natural language processing [17]. In the result section, we validate the tree distance metrics of the RNA secondary structures in their tree representations and use their distance matrices to classify the different types of RNAs. In addition, we use the 5-fold cross validation to verify distance calculations and classification algorithms.

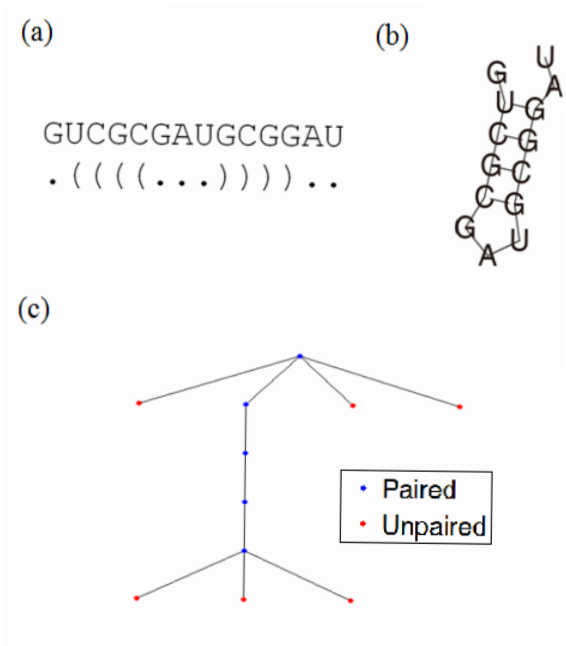

Figure 4.1: RNA secondary structure representations. (a) A sequence and its dot-Bracket Notation (DBN). (b) The secondary structure of (a). (c) A tree representation of (b).

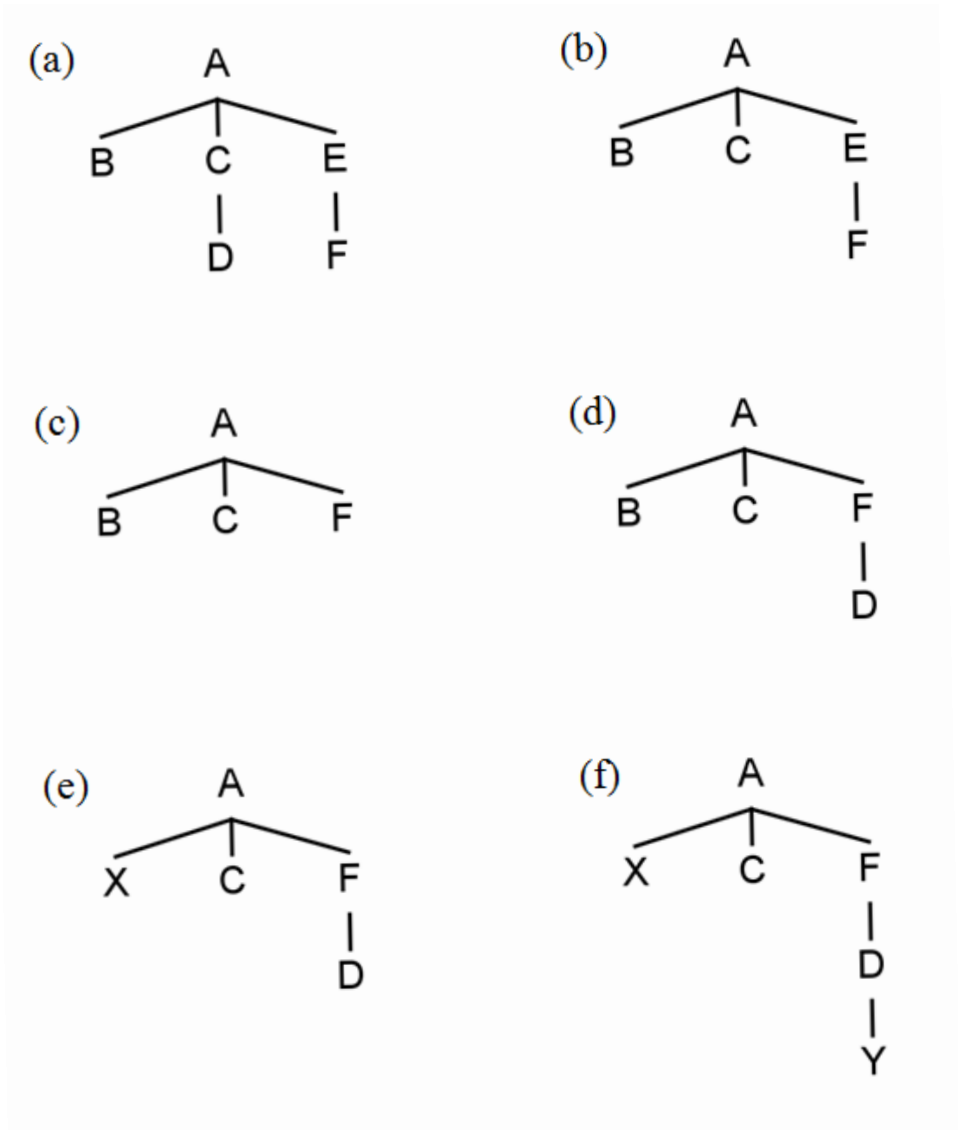

Figure 4.2: Tree edit operations. (a) Initial tree. (b) Deleting node D. (c) Deleting node E. (d) Inserting node D. (e) Modify node B to X. (f) Inserting node Y. This is the target tree (edit distance= 5).

4.2 Method

Conceptually, the algorithm to calculate the edit distance between two trees is similar to that to compute the edit distance between two strings. For each iteration of this algorithm, we start with a tree edited with one of the three fundamental operations: addition, deletion, or modification. The final edit distance will be the total number of such operations. Figure 4.2 shows an example of applying edit operations on a tree. We test the following algorithms: the full-resolution algorithm [37, 27], the coarse-grained, weighted coarse-grained [90], tree-translate algorithms, and the HITs algorithm [27]. In addition, a robust algorithm for tree edit distance (RTED) [81] is applied for comparisons to the Vienna RNA package [37] for full-resolution trees.

Additionally, we applied the kernel-based tree distance metric [17] to measure the distance between RNA structures. The distance $d(T_1, T_2)$ between tree T_1 and T_2 is calculated by the following equations 4.1–4.3.

$$K(T_1, T_2) = \sum_{n \in T_1} \sum_{m \in T_2} \sum_{i \in (s(T_1) \cup s(T_2))} I_i(n) I(m) = \sum_{n \in T_1} \sum_{m \in T_2} c(n, m) \tag{4.1}$$

$$c(n,m) = \begin{cases} 1, & \text{if } n \text{ and } m \text{ are unpaired nodes} \\ \prod_i^{PC(n)} (1 + c(C(n,i), C(m,i))), & \text{if } n \text{ and } m \text{ are paired nodes,} \\ & \text{and have the same } |PC(n)| \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$d(T_1, T_2) = 1 - \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) K(T_2, T_2)}} \tag{4.3}$$

where $s(T)$ is a set of subtrees which belong to tree T, $PC(n)$ is a set of paired children which have the node n as a root node, $C(n, i)$ is the i-th child of the

node n, and $I_i(n)$ is the 1 if a node n is the root node of subtree i, otherwise 0. $c(n, m)$ counts the number of the common structure between two trees whose root nodes are n and m.

For each of these algorithms, we make up the distance matrix for the whole sample data and classify the type of each sample RNA using the k-nearest neighbor (k-NN) classifier [9] with $k = 5$. The Vienna RNA package is used to calculate the distance. We use the k-NN method to classify the labels of the structures using the distance metric obtained from the edit distance. Each sample is from a microRNA or tRNA structure.

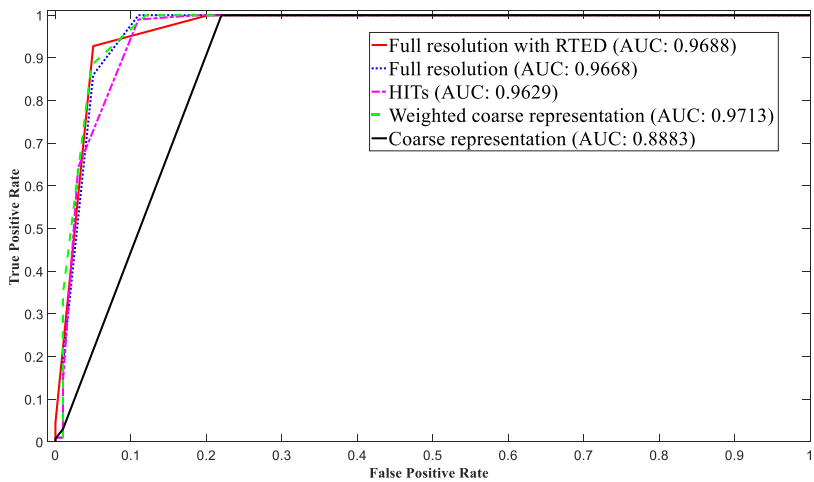

Figure 4.3: The ROC curves of different algorithms.

4.3 Result and discussion

The test data is from miRBase [53] and RNA STRAND [2] and is a mixture of miRNA and tRNA structures with the same proportion. For each dissimilarity measurement, Figure 4.3 and 4.4 show the receiver operating characteristic

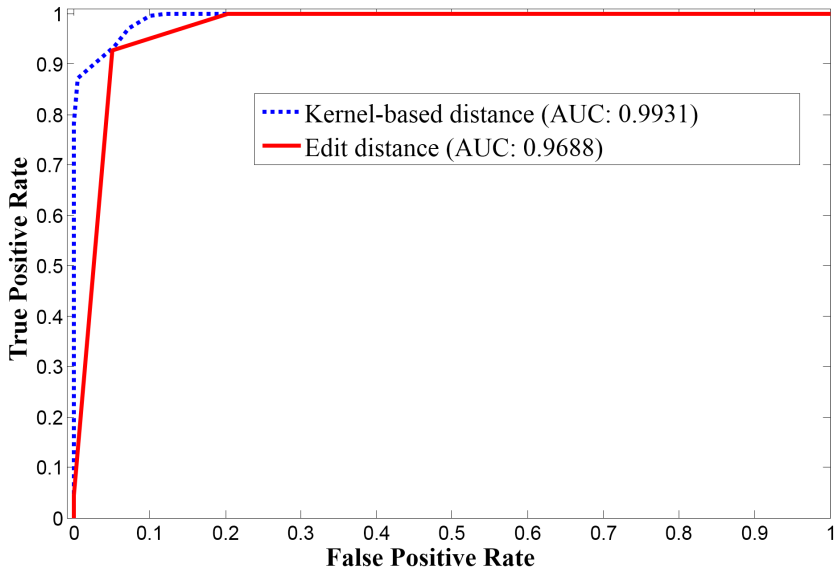

Figure 4.4: The ROC curves of kernel-based distance metric and RTED.

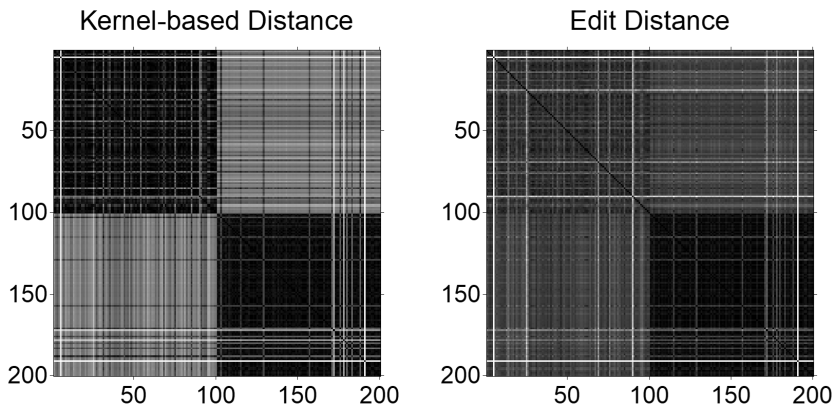

Figure 4.5: Pairwise distance matrix for comparing kernel-based distance and RTED. Each pixel in (x, y) represents the distance between sample RNAs x and y. Each pixel value is normalized into the range $[0, 1]$, and its brightness gets darker as their distance becomes smaller. The samples 1–100 are miRNA hairpins and the others are tRNAs.

(ROC) curve and the area under curve (AUC) values of different algorithms. Each result was acquired from the mean of the 5-fold cross validation. The k-NN classifier with the tree edit distance algorithm shows the effective classifying performance, which leads to successful separation of miRNA and tRNA structures. In terms of accuracy, the k-NN classifier predicted over 93% of samples correctly except for the coarse-grained tree.

For the kernel-based approach and the full resolution edit distance method, the differences between the average intra-class and inter-class distances are about 0.34 and 0.12, respectively. The result implies that the kernel-based approach may distinguish the miRNA hairpins from the tRNAs better than the edit distance methods. Figure 4.5 also supports the result. Based on the result presented in the chapter, we are planning to devise a distance metric for tree representations of RNA structures, which is efficient and accurate for a variety of machine-learning tasks in comparing RNA structures.

Chapter 5

Distributed deep learning for large scale data

5.1 Background

Deep Neural Network (DNN) has achieved remarkable performances in numerous realms of learning problems. In order to effectively train large DNN models, an extremely large amount of train data is required. Hence the need for a distributed system that manages such enormous computation tasks and data is inevitable [23]. Among the famous distributed pipelines, Apache Spark rapidly grew as one of the most attractive frameworks [109] owing to its fast in-memory data processing and high fault-tolerance. Consequently, the integration of deep learning and a Spark cluster has gained widespread attention and several frameworks such as SparkNet [75], CaffeOnSpark[1], and TensorFrames[2] were put forth. These attempts showcased the possibility of Spark becoming the basis for a powerful distributed deep learning framework. In

[1] `https://github.com/yahoo/CaffeOnSpark`
[2] `https://github.com/databricks/tensorframes`

terms of practical efficiency and scalability, however, there is significant scope for improvement.

Two key factors hinder the effective learning capability on commodity Spark clusters. The first is the synchronous operation characteristic, which is inherent in Spark itself. In general distributed learning algorithms, asynchronous approaches are known to be faster than simple synchronous alternatives [23, 83, 65]. That is, despite the higher number of learning steps required because of the discrepancy originating from overwrites, the asynchronous steps require relatively less time. A variant of such a synchronous method [14] aimed to tackle the slow learning step and achieve faster convergence by utilizing backup workers. However, increasing the number of worker nodes is not suited to commodity setups; moreover, the selective operations for backup workers are not supported by Spark. Therefore, asynchronous schemes such as *parameter server* are reasonable for commodity Spark clusters [23, 64].

The other obstacle is the lack of algorithmic considerations for reducing communication traffic. On the other hand, training DNNs is a communication-intensive task which demands frequent weight parameter exchanges [111]. Under commodity cluster settings where the network bandwidth is very constrained, the constant communication requests can lead to increased *communication overheads*. If the communication time outweighs batch computing time, achieving speed-up with the parallel training algorithms is unattainable. Most of the aforementioned Spark-based deep learning frameworks assume high-speed networks such as InfiniBand network or Amazon EC2, and thus operations might be slower on relatively low-cost clusters.

In this chapter, we propose DeepSpark, a new distributed deep learning framework for commodity Spark clusters with the objectives of overcoming

abovementioned limitations. We implemented the parameter server scheme [64] to run our asynchronous stochastic gradient descent (SGD) algorithm. To alleviate the communication overhead, we further designed an adaptive parameter dropping algorithm. DeepSpark combines Caffe [41] and TensorFlow [1], thus various training models from convolution based model to recurrent neural network models can be implemented on Spark pipelines. The proposed framework was evaluated through experiments on two representative types DNN models: image classification and sequence-to-sequence modeling. The proposed dropping algorithm improved the communication-to-computation time ratio up to 3× with rare or no loss. Our contributions are summarized as follows:

1. The reduced parameter exchange overhead through the adaptive dropping speeds up the DNN training.

2. The asynchronous iterative process on Spark provides fast DNN training.

3. The integration with popular deep learning libraries provides flexibility and accessibility to scale up various learning models.

5.2 Motivation

Apache Spark is an attractive platform for data-processing pipelines such as a database query processing. However, Spark RDD provides limited asynchronous operations between the master and the workers.

To addressing the disadvantages of Spark, we implemented a new asynchronous SGD solver with a custom parameter exchanger on the Spark environment. To improve asynchronous performance, we noticeably improved the EASGD algorithm [112] by considering adaptive parameter updates, thereby delivering faster convergence.

5.2.1 Parallelized SGD and its limitations

Our proposed framework utilizes a parallelized SGD algorithm, which is based on elastic averaging SGD (EASGD) [112]. The objective function of EASGD is as follows:

$$\underset{w_1,w_2,\cdots,w_P,\hat{w}}{\arg\min} \sum_{i=1}^{P} f(w_i; D_i) + \frac{\lambda}{2}\|w_i - \hat{w}\|^2 \tag{5.1}$$

where vectors w_i and $\hat{w}$ are the local parameters of the worker node i and the global parameter respectively; D is the training data, P is the number of worker nodes, and $f(w; D)$ is the cost function for local workers. Since the first term is identical to the sequential SGD algorithm, each worker node can train a model with the existing schemes such as momentum [76] and ADAM optimizer [47]. As the second term implies, EASGD affords elasticity when exchanging parameters instead of simply overwriting the global parameters and stabilizes the discrepancy provoked by asynchrony. The symmetric update between local and global parameters ensures the convergence of EASGD [112]. Moreover, EASGD relaxes the heavy communication loads of parallel SGD by introducing the time delay τ in the updating parameters. This feature makes the EASGD algorithm robust against time delay.

However, adjusting τ results in a trade-off between the communication workload and the amount of discrepancy penalty. In other words, a larger τ shortens the communication time; however, the number of iterations (or epochs) required to converge increased. Considering the penalty, the speed-up of asynchronous EASGD algorithm can be formulated as follows:

$$\texttt{Speedup} = \frac{T_{\texttt{seq}}}{T_{\texttt{par}}} = \frac{N_a(b)C(b)}{d(\tau, n)N_a(b)[C(b/n) + S/\tau]} \tag{5.2}$$

where b is the size of mini-batch, $C(b)$ is the computation time for the mini-batch, and $N_a(b)$ is the required number of iterations to reach target performance a; τ is the communication period and the following discrepancy penalty is denoted by $d(\tau, n)$; S is the communication overhead, which is affected by τ and the number of worker nodes n. As Eq. 5.2 suggests, varying the communication overhead S can influence the speed-up of parallel DNN training. In case of the commodity cluster ($S >> C(b)$), the synchronous schemes ($\tau = 1, d = 1$) like CaffeOnSpark cannot make any speed-up even with the infinite number of worker nodes, as shown in Figure 5.1. For sufficiently large τ, the training speed suffers from the growth of the discrepancy penalty $d(\tau, n)$.

5.3 Down-sizing DNN models

Several approaches have applied the concept of compressing DNN models to reduce the number of parameters. Studies such as [111, 15] aimed at reducing the communication requirements by exchanging error terms and activation vectors instead of the full gradient matrices. However, the scalability appears to be limited by increasing the mini-batch size and communication delay. Moreover, they are not compatible with weight parameter aggregation algorithms such as EASGD.

Lossy compression techniques have been proposed to transform the model structure and parameters into simpler formats [35, 39]. These techniques do not assume distributed training setup and merely focus on the effective compression of DNN models. Compression time or aggregation methods are not under their consideration, which makes them unsuited for parallel DNN training.

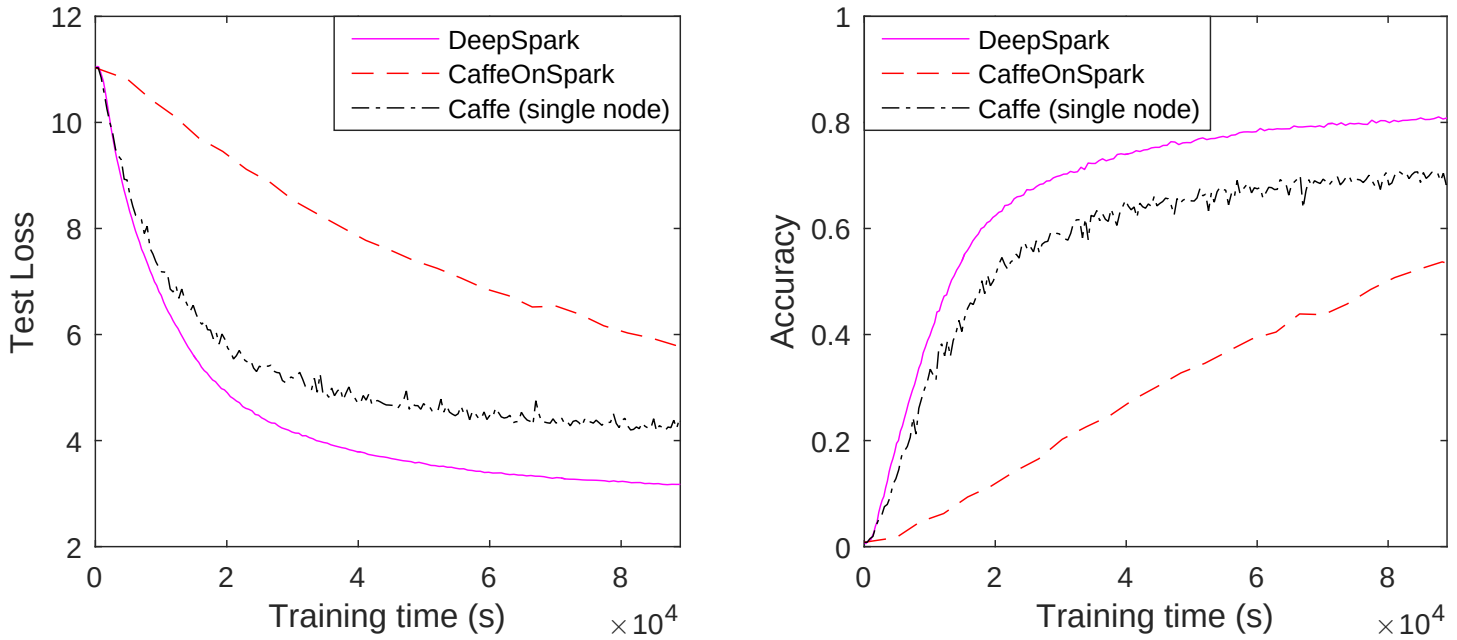

Figure 5.1: ImageNet training results on DeepSpark, CaffeOnSpark, and Caffe. Testing loss versus training time (left) and testing top-5 (right) accuracy versus training time. DeepSpark and CaffeOnSpark were tested on 16 executors. The experiments were performed without the distributed parameter exchanger and pruning scheme.

5.4 Apache Spark based approaches

Apache Hadoop YARN [96] and Spark [109] are cluster frameworks that allow large-scale data processing on commodity hardware. In Hadoop, dataset would be split into multiple blocks in Hadoop Distributed File System (HDFS) [92]. HDFS provides the overall control of these blocks and maintains fault tolerance. Hadoop YARN, which is the framework for resource management and job monitoring, is responsible for containers working in parallel.

Spark is the cluster computing engine running on YARN, Apache Mesos or EC2. The core of Spark is the in-memory distributed processing using resilient distributed dataset (RDD). RDD is the read-only collection of data partitioned across a set of machines. On RDD, parallel actions are performed to produce actual outcome or transformations can be applied to convert a certain RDD into other type of RDD. For further reuse, RDD can be cached in cluster memory, which prevents unnecessary storage I/O and thus accelerates data processing. Hadoop and Spark are originally designed for batch-synchronized analysis on large data. They are, however, less suited for tasks that requires asynchronous actions on parallel workers [108].

As the big data pipeline, Spark is the generalized successor of MapReduce which is a platform for distributing large amount of data to loosely coupled cheap computing nodes [109]. Till date, several research institutes, companies, and governments have built their own data processing pipeline on Apache Spark [78, 99, 70, 77, 103, 63]. Naturally, deep learning platforms have also been suggested on the Spark environment. In general, deep learning on Spark should be synchronous SGD for model aggregation, as is the case in other applications [75].

Asynchronous based learning approaches have shown faster convergence than pure synchronous approaches in previous works [23, 83, 65]. Although a variant of the synchronous method supported by backup worker nodes demonstrated more robust performance than the asynchronous approaches by resolving discrepancy [14], the backup worker configuration is not suitable to the commodity setup, moreover Spark does not support a selective barrier for backup workers. To overcome these limitations, Spark must support additional assistive asynchronous implementations such as parameter server [23, 64].

5.5 Proposed method

5.5.1 Pruning based parallel elastic averaging SGD

We propose the *pruning EASGD (P-EASGD)* algorithm that performs EASGD with pruned parameters. From the second term of Eq. 5.1, if w_i and $\hat{w}$ are sparse, the communication overhead will be alleviated by the reduced volume of parameters that must be exchanged. Eq. 5.3 represents the objective function of P-EASGD:

$$\underset{w_1,w_2,\cdots,w_P,\hat{w}}{\arg\min} \sum_{i=1}^{P} f(w_i; D_i) + \frac{\lambda}{2}\|\mathrm{diag}(\mu_i)(w_i - \hat{w})\|^2 \tag{5.3}$$

where μ_i is the masking vector of the worker node i that has the same length; it determines the parameter to be pruned before parameter exchange. $\mathrm{diag}(\mu_i)$ is the diagonal matrix whose diagonal entries identical to the elements of μ_i. Each element of μ_i is a random variable that its outcome is either 0 or 1. Consequently, the masking vector transforms the exchange parameter vector $(w_i - \hat{w})$ into a sparse vector form [35]. The proposed pruning method does

not affect the assumed symmetry of update, therefore, P-EASGD is expected to maintain convergence and stability of the EASGD algorithm. The only difference is that the effective update step size of gradient may be decreased.

5.5.2 Parameter update rule

The P-EASGD update rule is acquired by calculating gradients from Eq. 5.3 with respect to w_i and $\hat{w}$. The rules can be formulated as follows:

$$\begin{aligned} w_i^{t+1} &= w_i^t - \eta g(w_i^t; D_i) - \alpha \,\mathrm{diag}(\mu_i^t)(w_i^t - \hat{w}^t) \\ \hat{w}^{t+1} &= \hat{w}^t + \alpha \sum_{i=1}^{P} \mathrm{diag}(\mu_i^t)(w_i^t - \hat{w}^t) \end{aligned} \tag{5.4}$$

where w_i^t and $\hat{w}^t$ are local and global model parameter respectively at iteration t, and η is the learning rate. μ_i is the pruning mask vector generated from the worker node i at every iteration, and is drawn from Bernoulli distribution with probability $1 - \rho$. Each element of μ_i is set to zero with probability ρ, otherwise 1 with probability $(1 - \rho)$. $\alpha = \eta\lambda/P$ is the moving rate for EASGD update [112] and determines the elastic force between the local and the central parameters; it determines the extent of the training that is affected by the distance between those parameters. The entire process of the P-EASGD algorithm is described in Algorithm 1.

When each worker node sends weight parameters w_i, the parameters are transformed into the sparse vector of $\mathrm{diag}(\mu_i)w_i$, as described in Figure 5.2. A parameter exchanger also returns $\mathrm{diag}(\mu_i^t)\hat{w}^t$ in the same form, and then, they update w_i^{t+1} and $\hat{w}^{t+1}$. In our implementation, we assigned 8 bits to the relative index, and a value of the parameter was stored as a 32-bit single precision floating point type. Consequently, we can reduce the original parameter size of

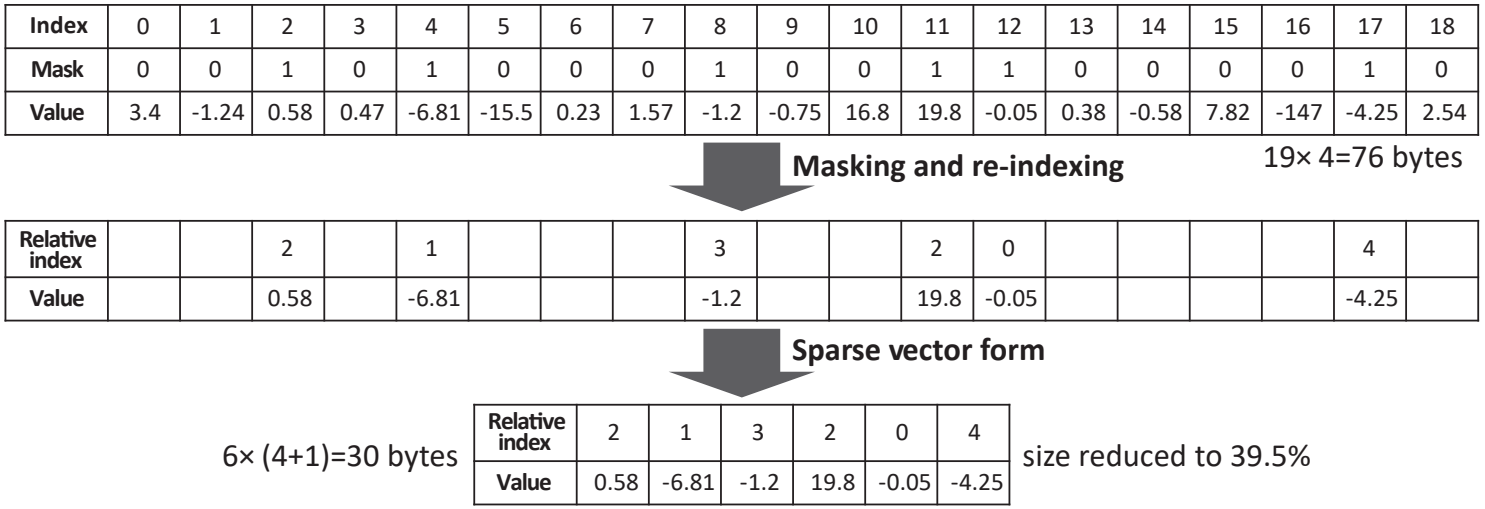

Figure 5.2: An illustration of sparse representation scheme. The pruning mask μ_i is drawn from $Bernoulli(1-\rho)$, and then the 8-bit relative indices are attached to the sampled values.

Algorithm 1 Pseudo-procedure of P-EASGD training for a worker node k

Input: $i_{max}, i_p, \rho, \alpha$ ▷ i_{max}: # of iterations for the whole training, i_p: # of iterations for the pre-training, ρ: pruning ratio, and α: moving rate for the elastic averaging update.
Output: $w_k, \hat{w}$ ▷ w_k: local model parameters, $\hat{w}$: global model parameters

InitializeParameters($\hat{w}$) ▷ Initialize the model parameters
$w_k \leftarrow \hat{w}$
▷ pre-training phase
for $i = 1$ to i_p **do**
  $\Delta w_k \leftarrow$ ComputeGradient(w_k)
  if $i \mod \tau == 0$ **then**
    $w_k \leftarrow w_k - \alpha(w_k - \hat{w})$
    $\hat{w} \leftarrow \hat{w} + \alpha(w_k - \hat{w})$
  end if
  $w_k \leftarrow w_k + \Delta w_k$
end for
▷ pruning phase
for i to i_{max} **do**
  $\Delta x_k \leftarrow$ ComputeGradient(w_k)
  if $i \mod \tau == 0$ **then**
    Random Sample μ mask: $\mu \sim Bernoulli(1-\rho)$
    $w_k \leftarrow w_k - \alpha\,\mathrm{diag}(\mu)(w_k - \hat{w})$
    $\hat{w} \leftarrow \hat{w} + \alpha\,\mathrm{diag}(\mu)(w_k - \hat{w})$
  end if
  $w_k \leftarrow w_k + \Delta w_k$
end for

$4n$ bytes for parameter exchange to $5n(1-\rho)$ bytes in average, for n model parameters. We used the sparse form for the parameter exchange; therefore, the learning procedure and trained results are compatible with existing optimizers and DNN models, unlike DeepCompression [35].

The random pruning process raises concerns about intensified inaccuracy during training. We observed that training with P-EASGD from scratch generates additional errors and exacerbates the convergence in the early stages, as illustrated in Figure 5.12. The error falls into reasonable level as training progresses. Hence, we planned a two-step training strategy; training with the intended pruning rate after pre-training with zero pruning rate. The related experimental results and interpretation are described is shown in Section5.8.1.

5.6 Implementations

5.6.1 Structure overview

Our DeepSpark framework consists of three main parts, namely, Apache Spark, a parameter exchanger for asynchronous SGD, and the Caffe and TensorFlow software, as shown in Figure 5.3. Apache Spark manages workers and available resources assigned by a resource manager.

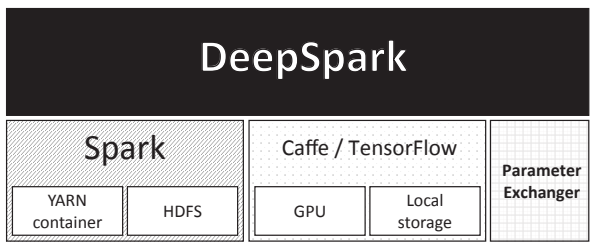

Figure 5.3: Stack diagram of DeepSpark architecture.

Figure 5.4 depicts how the Spark workflow progresses for asynchronous SGD, and we provide more detailed descriptions in Sections 5.6.2 and 5.6.3. Subsequently, we explain the parameter exchanger and asynchronous SGD process exploiting Spark in Sections 5.6.3, 5.6.4 and Figure 5.5. We clarify how to integrate Caffe and TensorFlow as our SGD computing engine with Spark using Java Native Access (JNA)[3] interface in Section 5.6.5. The overall procedure of our framework is summarized in Algorithm 1. We assumed that the framework runs on Hadoop YARN [96] and HDFS [92] in the following explanation of our workflow.

[3] `https://github.com/java-native-access/jna`

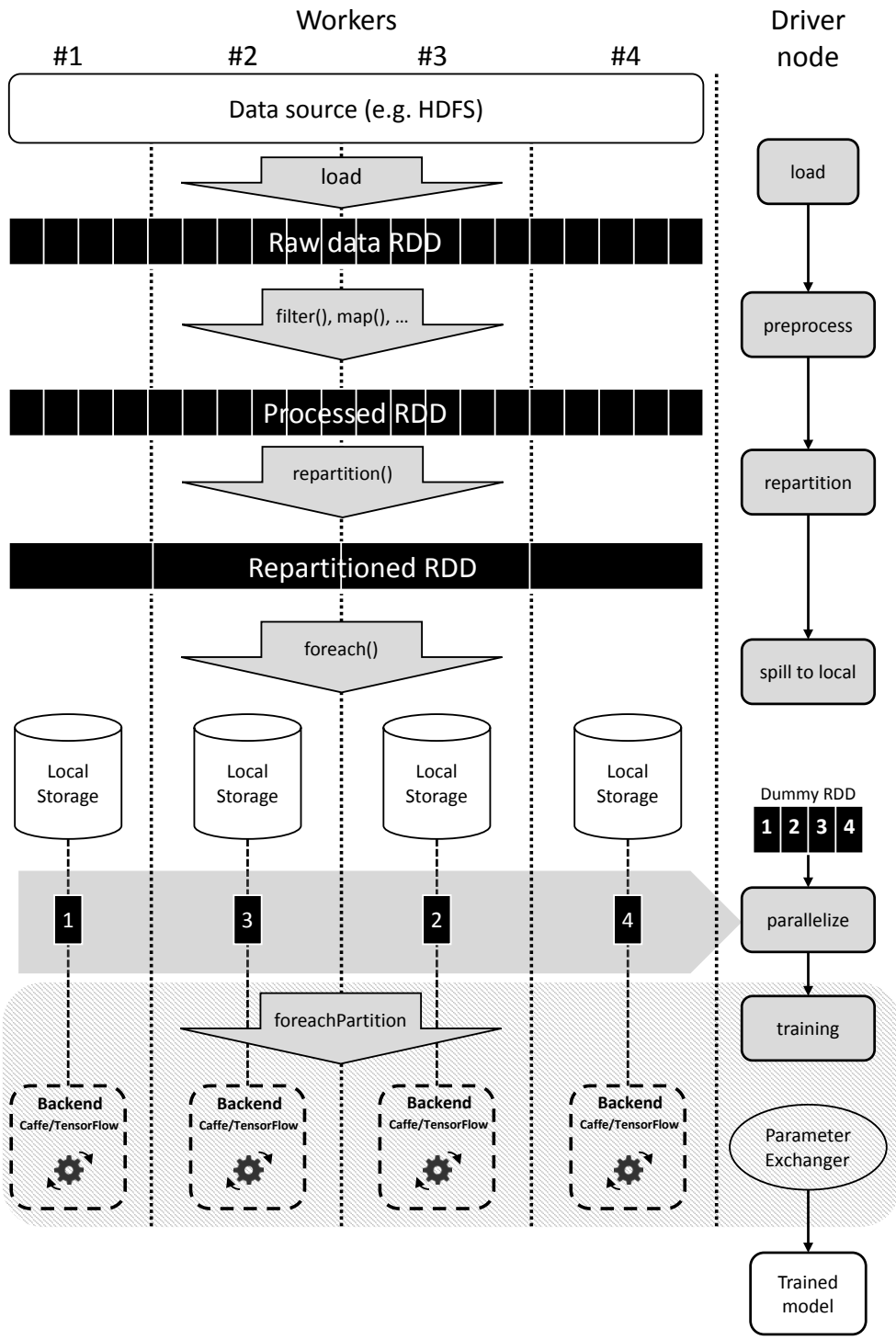

Figure 5.4: Spark workflow of DeepSpark learning process.

5.6.2 Implementations on Spark

In this section, we explain the proposed framework's distributed workflow from the data preparation to asynchronous SGD, which is corresponding to load and spilling phase in Figure 5.4. Given that the proposed framework is running on top of the Spark framework, it needs to load and transform the raw data in the form of Spark RDD [108].

The first step of the training is to create RDD for training and inference. We defined a container class, which stores label information and corresponding data for each data sample. The data specific loader then creates the RDD of this data container class, which is followed by the preprocessing phase. In the preprocessing phase, data containers would be connected to the preprocessing pipeline such as filtering, mapping, transforming, or shuffling. The processed RDD repartitioning is then performed to match the number of partitions to the number of worker executors.

Caffe and TensorFlow, the actual computing engine, however, cannot directly access RDD. In our framework, the entire dataset is distributed across all workers, and each worker can cache or convert its own parts of dataset into LMDB[4] file format if the data are relatively larger than the memory size. For a relatively small dataset, the RDD `foreachPartition` action is executed, where every data partition is loaded in local worker's memory as a form of Java List. These data then become available by neural network model of Caffe or TensorFlow, directly.

The other approach to feed the data into the native codes is spilling the dataset on a worker node's storage. For a large dataset that is difficult to hold in the physical memory, the RDD `foreach` operation is performed, and each

[4] `http://lmdb.readthedocs.org/en/release/`

data partition is converted to LMDB file format and stored in the temporary local repository of the node it belongs to. Once the LMDB files are created, Caffe automatically computes data dimension and finally completes the neural network model parameter. In addition, we also developed miscellaneous functions such as data feeder for TensorFlow. We used LMDB JNI[5] to manipulate LMDB on Spark.

5.6.3 Asynchronous EASGD operation

Inherently, Spark does not support step-wise asynchronous operations for asynchronous SGD updates. We adopt the method that exploits Spark RDD operations to overcome the limitation of Spark. The dummy RDD represented in Figure 5.4 can mimic the asynchrony.

Once the LMDB local repository for each worker has been prepared, the dummy RDD is created and distributed across every worker. These dummy data have an important role in launching the distributed action (*i.e.*, parallel model training is performed). Although the explicit dependency between spilling and training steps is nonexistent at the code level, each worker node would be guided to launch the training process with a spilled dataset by the dummy RDD. This exploits the property of Spark that the Spark scheduler reuses the pre-existing worker node session. The size of the dummy RDD is explicitly set to the number of workers for full parallel operation, and the `foreachParition` action is executed on this dummy RDD. Inside the `foreachPartition` process, each worker can use the local data repository that has been created in the previous job and starts the training step.

During the training process, the Spark driver program serves as a central

[5] `https://github.com/chirino/lmdbjni`

parameter exchanger, which performs an asynchronous EASGD update. At the initial step, the driver node broadcasts its network address and neural network setup files to all workers. Workers then create their own models and start training using broadcasted data.

5.6.4 Parameter exchanger

The parameter exchanger is our implementation of parameter server concepts, which is essential for asynchronous update. In our framework, the application driver node serves as the coordinator to enable worker nodes to communicate their parameters to other workers asynchronously. Figure 5.5 shows the outline of the learning cycle with the parameter exchanger in each worker.

When multiple-parameter exchange requests from worker nodes exist, a thread pool is implemented to handle the requests at the same time. For each connection request, the thread pool allocates the pre-created threads that process the parameter-exchange requests. The size of the thread pool is fixed in the program , and we set this up to eight threads because of limited memory and network bandwidth. If the number of requests exceeds the size, the unallocated requests wait in a queue until the preceding requests are completed as shown in Figure 5.5(a).

Exchange threads asynchronously access and update the neural net model in the parameter exchanger based on the EASGD algorithm. In Figure 5.5(b), given that it is a lock-free system, parameters can be overwritten by simultaneous updates. Nevertheless, training results are accumulated successfully, as proven in [83]. After the parameter exchange action, each worker returns to the SGD phase to explore the parameter space asynchronously as shown in Figure 5.5(c).

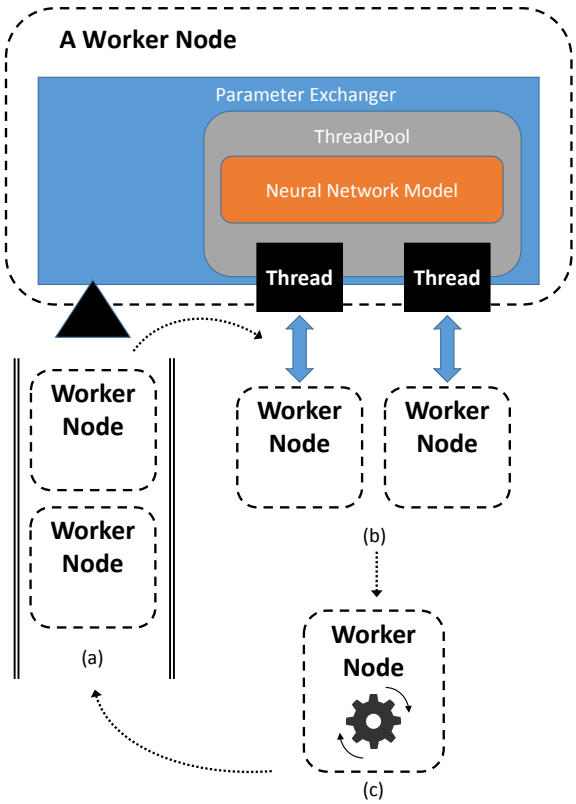

Figure 5.5: Learning process with parameter exchanger in each worker. (a) Worker nodes that want to exchange the parameters are waiting in a queue until an available thread appears. (b) Exchanger threads take care of the worker request. In this example, there are two exchanger threads in the pool. (c) After exchange, a worker node performs its SGD process during the communication period τ.

Coordinator Table (Spark Driver)

Worker	Offset	Length
Worker1	[0, 0, 0]	[3, 3, 3]
Worker2	[3, 3, 3]	[3, 3, 3]
Worker3	[6, 6, 6]	[3, 3, 3]

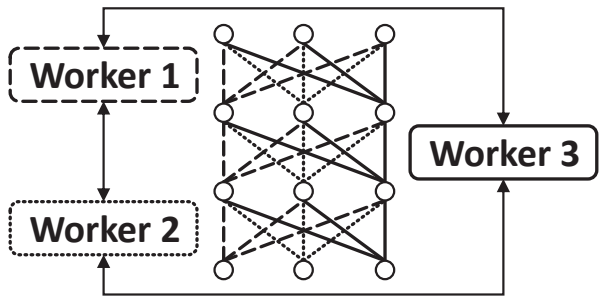

Figure 5.6: Example of the distributed parameter exchanger. Each worker node takes responsibility for a part of the global parameters $\hat{w}$, respectively. During the training, each worker node communicates with the others according to the coordinator table.

However, when each pruning process is performed on a single parameter server, it would cause further overhead to that machine because of the limited network bandwidth and system resources. This issue was resolved by distributing the role of parameter servers to multiple worker nodes. Each worker node simultaneously performs learning and parameter exchange by multi-threading, and takes the responsibility for partitioning of the global parameters $\hat{w}$. The Spark driver node only provides the coordinator table on which the offset and length of the model parameter partition are written. Each worker node exchanges the partition of its own w_i with the corresponding part of $\hat{w}$ in the others, by referring to the table from the driver node. Figure 5.6 shows our distributed parameter exchanger scheme.

5.6.5 Backend engine

Each worker node in DeepSpark can use either the Caffe library and TensorFlow for the GPU-accelerated backend engine. However, Spark application is written in Java, Scala, and Python, which cannot use the native backend engines directly in the source code level. We implemented our code with JNA so that Spark executors can reference the native library of Caffe and TensorFlow. We defined an additional wrapper for solvers in Caffe to perform an atomic iteration action, acquire current trained parameters, and modify. This custom wrapper provides an interface to control the Caffe library for the DeepSpark application. Therefore, current Caffe models can be used in a distributed environment without changing the Caffe network specifications numerous times. In case of TensorFlow, we also wrote an additional Java wrapper for TensorFlow C++ native library. We can run a Tensorflow session on the wrapper with the operational graph which was obtained from model description written in Python.

5.7 Experimental results

We built a commodity cluster environment for validating the performance of our algorithm. Each worker node consisted of an Intel Core i7-4790 CPU, 16GB memory, and an NVIDIA GeForce 970 GTX GPU device with 4GB memory. Given that this specific model of GPU had poor access to the memory region above 3.5GB[6], we restricted the use of memory to under 3.5GB for computing the local gradient step. The nodes were interconnected via Gigabit Ethernet, which had 1Gbps of network bandwidth. As a software environment, Apache

[6] `https://blogs.nvidia.com/blog/2015/02/24/gtx-970/`

Spark 1.6.1, Hadoop 2.6.1, NVIDIA driver 352.63, CUDA 7.5, and CuDNN 4 were installed on all worker nodes. Caffe and TensorFlow were adopted as back-end computing engines and compiled from the source code of which the version were 1.0.0-rc3 and 1.0.1, respectively. In this section, describe the training of two different CNN based models and a RNN based model, and compared the learning curve of P-EASGD to EASGD as a baseline.

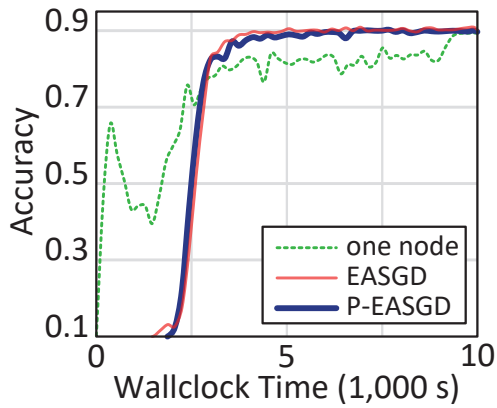

Figure 5.7: Learning curve for ResNet-56 with CIFAR-10 dataset. The validation accuracy relative to time is shown (# of node=4, $\rho = 0.75$, $\tau = 10$).

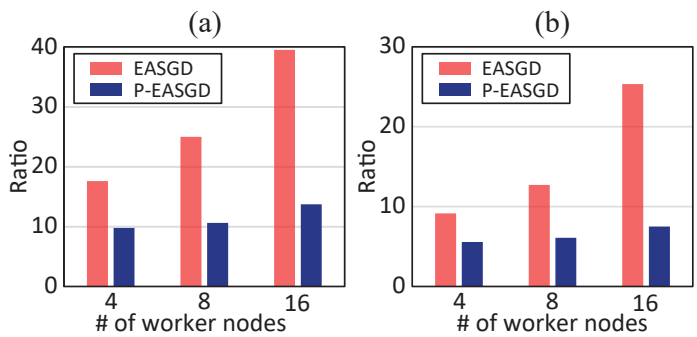

Figure 5.8: Communication-to-computation time ratio of EASGD and P-EASGD. Each result is about (a) AlexNet and (b) sequence-to-sequence model. $\rho = 0.75$ was set for each model.

Table 5.1: Average communication and computation time for training AlexNet and sequence-to-sequence model.

	AlexNet		**seq2seq**	
	Communication	Computation	Communication	Computation
EASGD (16 nodes)	10,368ms	262ms	14,167ms	559ms
EASGD (8 nodes)	6,635ms	265ms	8,154ms	641ms
EASGD (4 nodes)	4,623ms	262ms	5,827ms	636ms
P-EASGD (16 nodes)	3,642ms	265ms	4,668ms	622ms
P-EASGD (8 nodes)	2,828ms	266ms	3,896ms	638ms
P-EASGD (4 nodes)	2,598ms	265ms	3,451ms	619ms
Single Machine	-	268ms	-	1,436ms

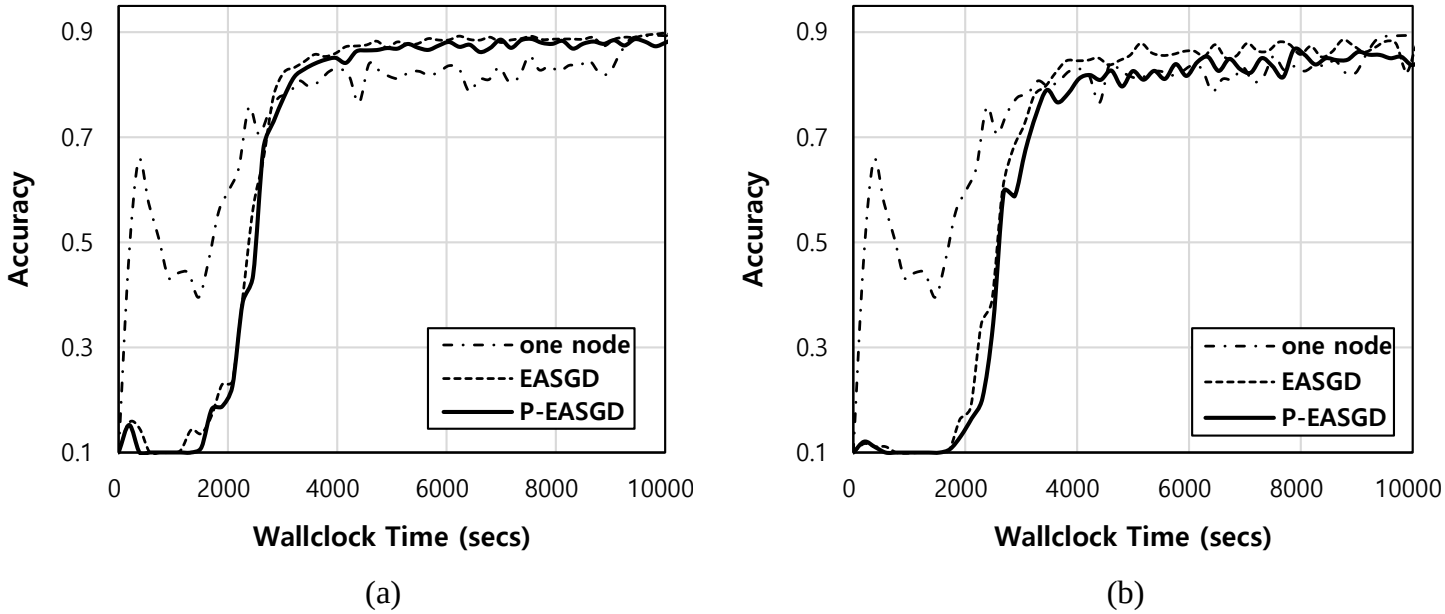

Figure 5.9: Learning curve for ResNet-56 with CIFAR-10 dataset. The validation accuracy versus time is shown for (a) 8 nodes and (b) 16 nodes ($\rho = 0.75$, $\tau = 10$, and $\alpha = 0.2$).

5.7.1 Image classification

We examined the ResNet [36] and AlexNet [54] models for image classification. The ResNet-56 model for classifying CIFAR-10 consisted of 56 layers[7]. As most layers of the ResNet model were convolution layers, each layer had a small number of learning parameters compared to the number of its layers ($\sim$ 3.5MB). For this reason, the communication burden was relatively smaller than the batch computation time for training the ResNet-56 model, and acceleration of convergence time using P-EASGD not expected. Learning rate η was set to 0.1 at the beginning for all cases. ADAM optimizer was employed for training on a single node, and we reduced η to one tenth after 25,000 iterations.

In the experiments of parallel training for ResNet-56 with EASGD, the momentum SGD optimizer with a momentum factor $\mu = 0.9$, moving rate $\alpha = 0.1$, and communication period $\tau = 10$ were used. For P-EASGD, $\alpha = 0.2$, pre-training iteration $i_p = 500$ and pruning rate $\rho = 0.75$ were given. l_2-regularization parameter $\lambda = 0.0005$ was set for all the CIFAR-10 experiments. The accuracy curves on four nodes are shown in Figure 5.7. We conducted similar experiments on 8 and 16 nodes, and the results are shown in Figure 5.9. P-EASGD showed similar convergence to that of vanilla EASGD. During all ResNet-56 experiments, a worker node spent 363ms computing the gradient step on average. For EASGD and P-EASGD, a worker node consumed average 110 and 100 ms per exchange parameters. Compared to EASGD, as expected, pruning did not significantly benefit P-EASGD in terms of the communication time.

For image classification using AlexNet[8], we trained AlexNet with the ILSVRC

[7] `https://github.com/yihui-he/resnet-cifar10-caffe/tree/master/resnet-56`
[8] `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`

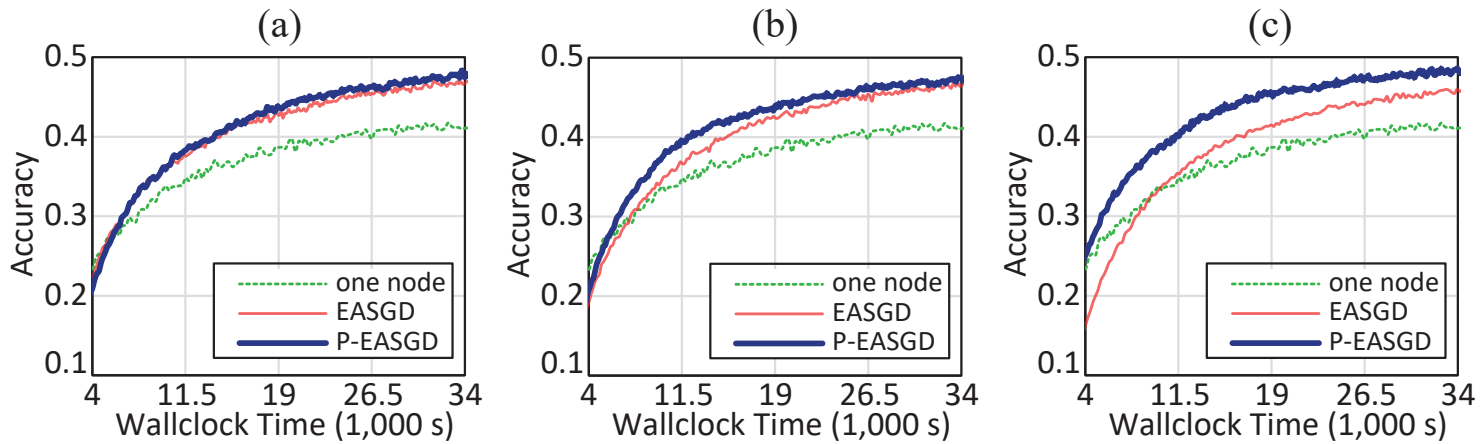

Figure 5.10: Learning curve for AlexNet with ImageNet dataset. The validation accuracy relative to time is shown. The results were trained on (a) 4 nodes, (b) 8 nodes, and (c) 16 nodes.

2012 ImageNet data set. In addition to the output layer, AlexNet has two inner product layers. It also has relatively more learning parameters ($\sim$ 243.9MB) than ResNet-56 and we expected P-EASGD to take advantage of pruning. For validating the trained model, we sampled 5K images from 50K of the validation set to shorten the validation time. ADAM optimizer was used to calculate gradient step for every single and parallel cases. As regards the hyperparameters, we fixed 128 as the batch size, $\eta = 10^{-4}$, $\lambda = 5.0 \times 10^5$, $\tau = 40$, $\alpha = 0.1$, $\rho = 0.75$, and i_p =3,000. Only the P-EASGD experiment on 16 nodes had different $\alpha = 0.4$. The reason why we set higher α for P-EASGD than EASGD is that we compensated the diminished step size caused by P-EASGD. We also repeat the same experiments for 4, 8, and 16 worker nodes to evaluate the scalability. The accuracy curves are shown in Figure 5.10. During all the AlexNet experiments, the average computing time was kept to near 262ms, however, the average communication time was increased as the number of worker nodes grew. We showed the ratio of communication time to batch computing time in Figure 5.8, and described the details in Table 5.1. Compared to EASGD, P-EASGD reduced the ratio and the communication time by up to 65.3% and 64.9%, respectively.

5.7.2 Sequence-to-sequence modeling

We also evaluated our approach in the sequence-to-sequence model for neural machine translation. To train neural machine translation models for English to French translation, we employed the English-French parallel dataset from the shared translation task of WMT'15 [10]. We used newstest2013 set to validate trained models. Our models consisted of four layers of gated recurrent units [16], each of which having 512 hidden units. The vocabulary size is limited

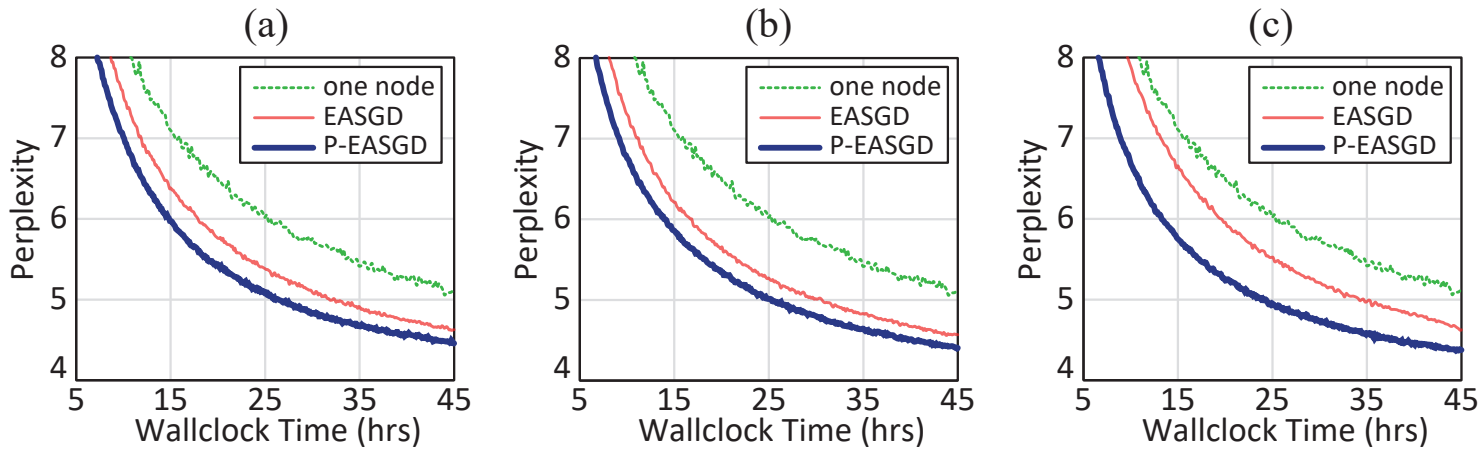

Figure 5.11: Learning curve for training sequence-to-sequence model. The validation error (perplexity) relative to time is shown. The results were trained on (a) 4 nodes, (b) 8 nodes, and (c) 16 nodes.

to 40K for each language. For the decoder network, the attention mechanism of Bahdanau et al. [4] was employed. The sampled softmax loss of 512 sample was used to handle large output vocabulary [40]. Lastly, we used Adam for optimization. The RNN models had many learning parameters over 300MB, and thus we expected the benefit from pruning like AlexNet.

For all experiments, we set $\eta = 0.0002$, 64 as the batch size, $\alpha = 0.1$, $\tau = 20$, $\rho = 0.75$, and i_p =1,000. The validation error curves are exhibited in Figure 5.11. Like result of AlexNet, the convergence of $P = 16$ was slower than that of $P = \{4, 8\}$ in the EASGD setting because of communication time. Figure 5.8 shows the ratio of communication time to batch computing time is displayed with the result of AlexNet, and the details are described in Table 5.1. Comparing to EASGD, P-EASGD reduced the ratio and the communication time by up to 70.4% and 67.1%, respectively.

5.8 Discussion

5.8.1 Justification of pre-training and pruning

Reasoning that enables pruning in the parameter exchange of distributed learning, we evaluate the mean squared errors between w_i and $\hat{w}$ with random mask μ_i in EASGD model for CIFAR-10 training. We simulated the error relative to the number of iterations by generating μ_i for 100 times, with respect to pruning rate ρ. The results are shown Figure 5.12. The metric for the error is described in Eq. 5.5:

$$\texttt{error} = \frac{1}{n}\|\text{diag}(\mathbf{1} - \mu_i)(x_i - \hat{x})\|^2 \tag{5.5}$$

where n is the number of parameters, and μ_i is the random pruning mask of worker node i. As we expected, the error started from high and then dropped sharply in early steps for all pruning rates. These results justify our assumption for pre-training. Although we expected pre-training to be critical to fast, high convergence, Figure 5.13 shows that the effect of pre-training was not significant in CIFAR-10 training. In future work, theoretic analysis and further experiments will be required to prove effectiveness of pre-training.

DropConnect [98] may have a similar concept to our method in that both of them prune some of the learning parameters. In DropConnect, masking vectors are drawn from each input sample for the regularization effect. P-EASGD, however, has different purpose and target from DropConnect. Generating μ_i per communication step only for sparsity, P-EASGD does not affect local parameters. Since both of techniques affect different terms independently, we can use both of them together for training.

5.8.2 Time and iteration efficiency

Figure 5.8 shows the communication to computation time ratio. The batch computation time refers to the time taken to compute gradient and update w_i. On the other hand, the communication time included transferring and updating time for w_i and $\hat{w}$. In the P-EASGD setting, the pruning and reconstruction time were added to the communication time. For the AlexNet and sequence-to-sequence model, the time spent for the communication took up to 39×. This is a major factor that significantly reduces the advantages of distributed training. Under the same condition, P-EASGD with ($\rho = 0.75$), however, reduced the communication overhead to less than a half. While the time efficiency was improved drastically as shown in Figs. 5.16 and 5.17, the conver-

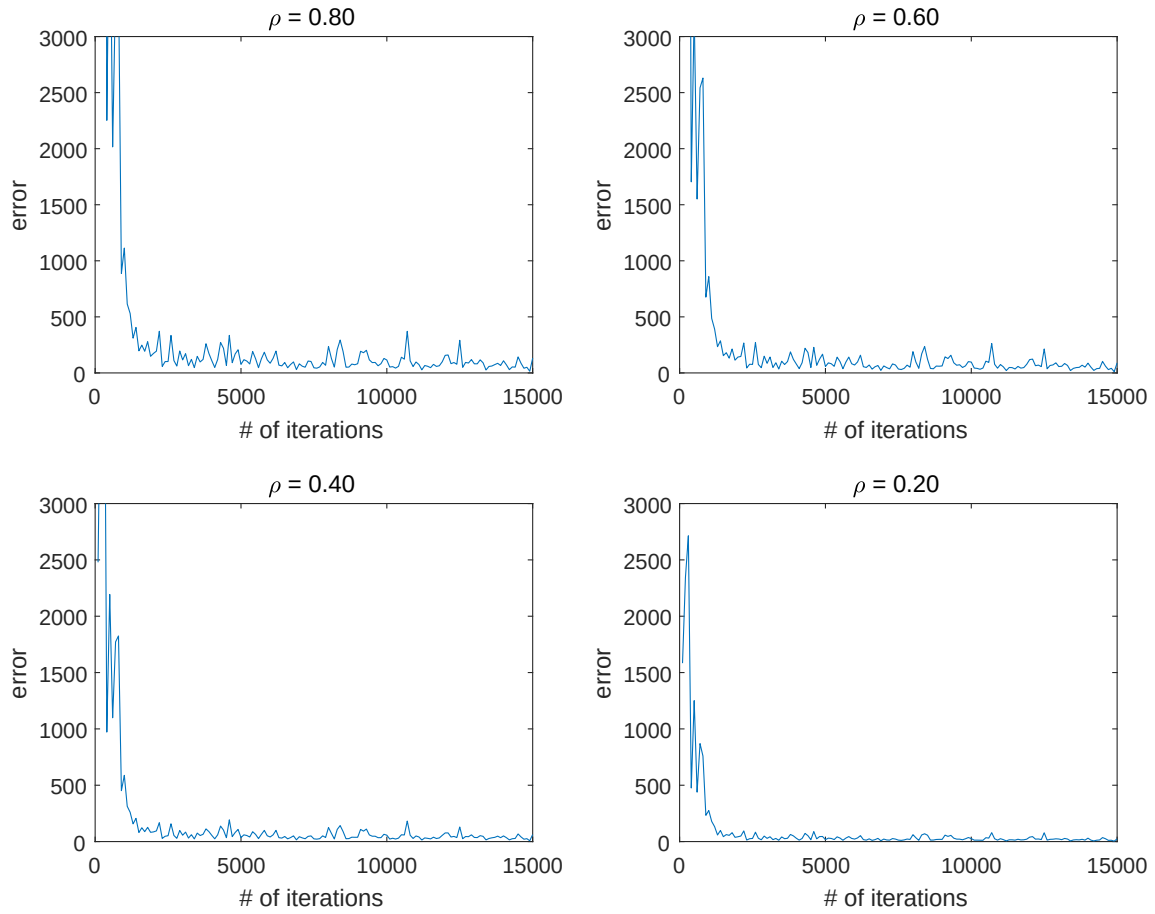

Figure 5.12: MSE versus the number of iterations for different pruning rate ρ (ResNet-56 for CIFAR-10)

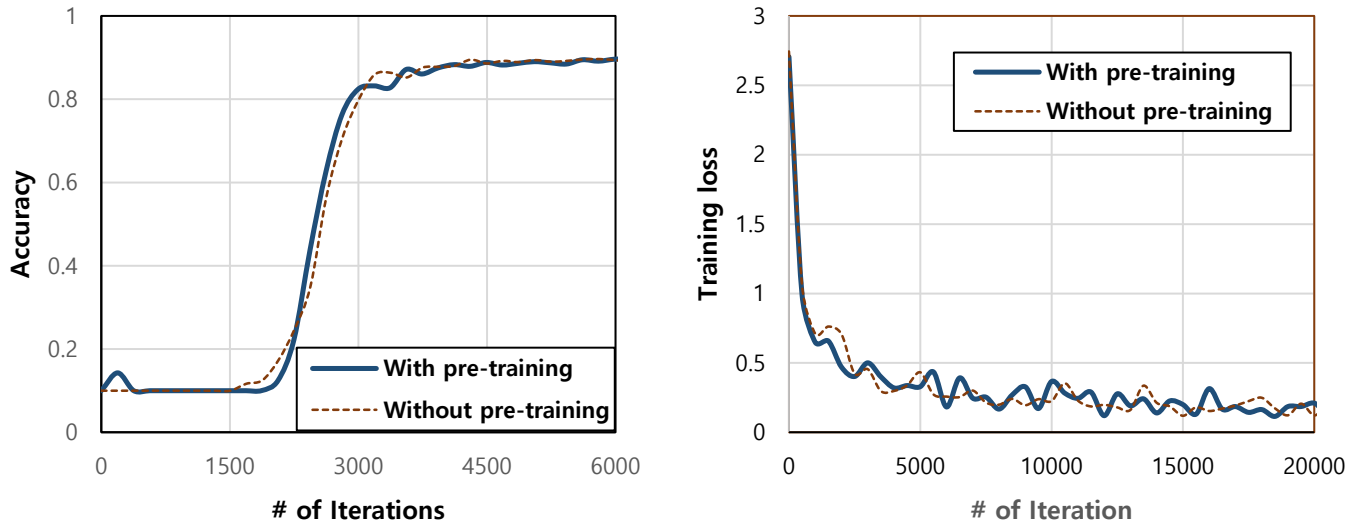

Figure 5.13: CIFAR-10 training result with pre-training and without pre-training on 4 nodes. Each graph shows (a) validation accuracy and (b) training loss versus the number of iterations. ($\alpha = 0.2$ and $\rho = 0.75$)

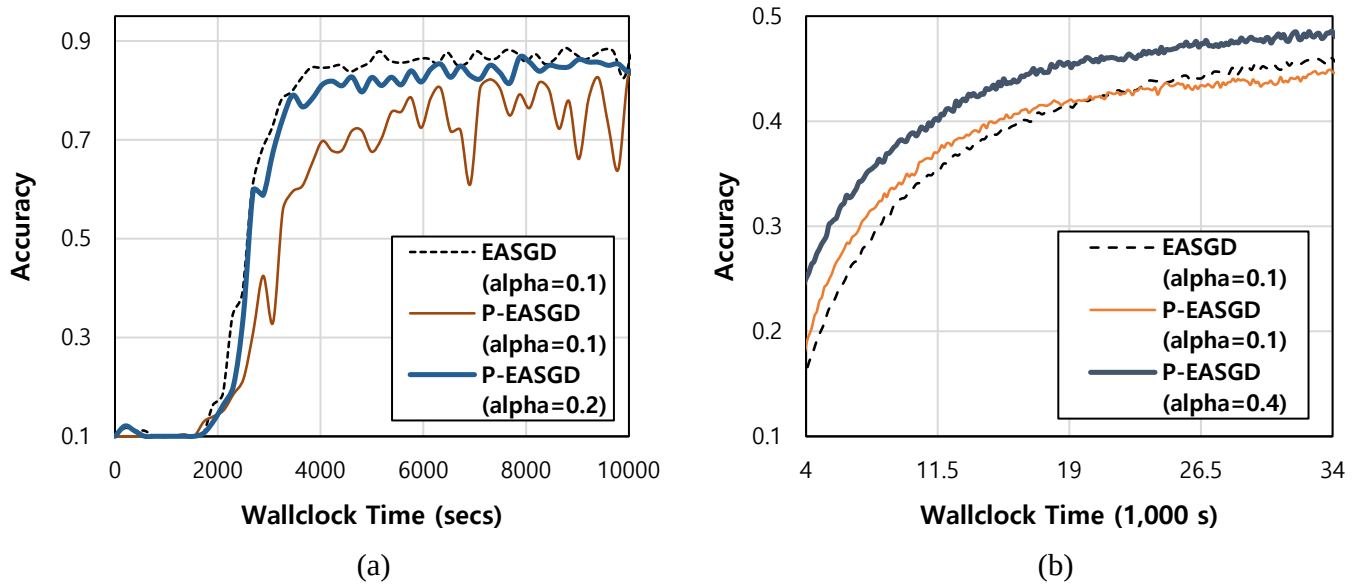

Figure 5.14: Validation accuracy versus wallclock time curves of training (a) ResNet-56 and (b) AlexNet on 16 nodes. P-EASGD does not catch up with the convergence of EASGD, at $\alpha = 0.1$.

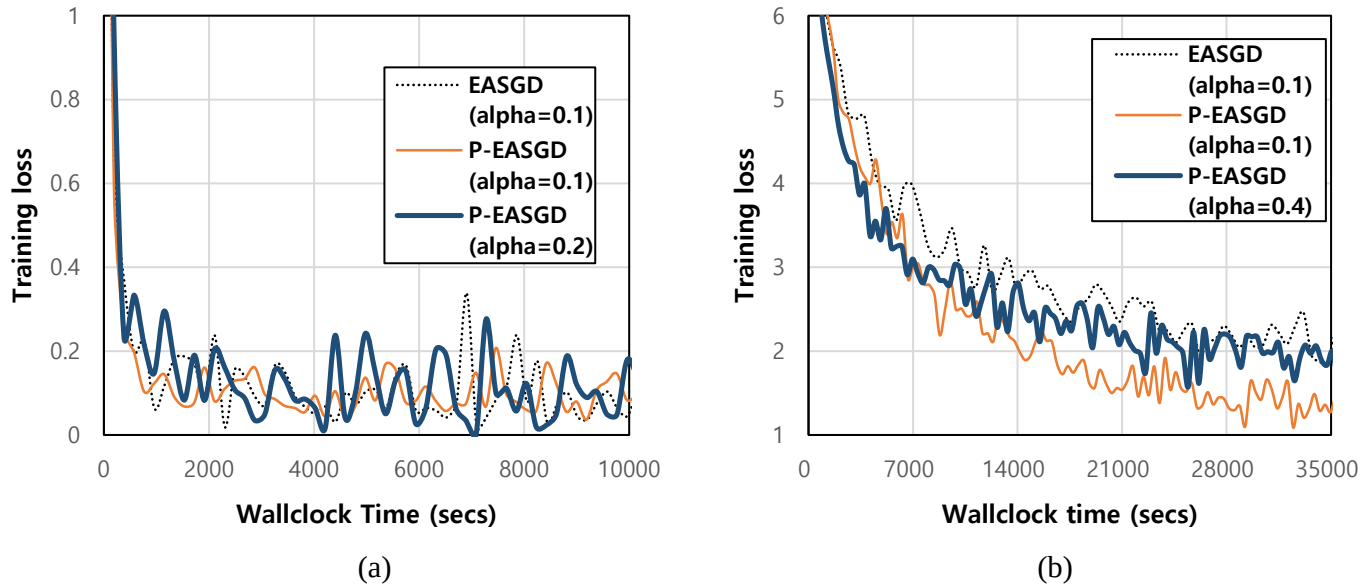

Figure 5.15: Worker node's local training loss versus wallclock time curves of (a) ResNet-56 and (b) AlexNet on 16 nodes. In training loss, it seems that P-EASGD converge faster at $\alpha = 0.1$.

gence difference between P-EASGD and EASGD was little. Consequently, the improvement of performance could be achieved because the gain in communication time, which outweighed the resulting degradation in training efficiency. This fact leaded to improve the scalability under the commodity environments. The P-EASGD also restrained the growth of communication-to-computation ratio more effective than the EASGD setting as shown in Figure 5.8.

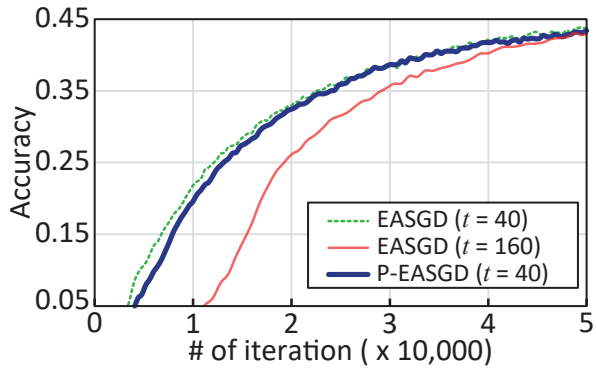

Figure 5.16: Validation accuracy versus iterations for training AlexNet model on eight nodes. t means the communication period. ($\rho = 0.75$)

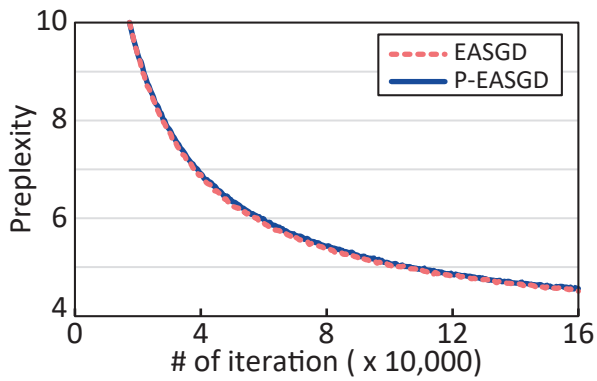

Figure 5.17: Validation error versus iterations for training sequence-to-sequence model on eight nodes. ($\tau = 20$, $\rho = 0.75$)

The batch computation time for sequence-to-sequence model on a single node was about twice larger than other distributed settings. The main reason

was that the training data was too large to be loaded on the single node memory, and some of the training data was spilled to the swap area in the local disk. The distributed setting moderated the memory usage for holding training data, while data spilling led to the increased data access overhead in a single node. As a result, we verified the benefit of distributed training in terms of resource usage with the aid of data parallelism.

One additional experiment was conducted to determine the better strategy by, exchanging all the parameters with longer τ or sampled parameters with short τ. Intuitively, considering each element of learning parameter w, pruning with a probability ρ implies that the individual communication period of each element follows a geometric distribution with success probability $p = 1 - \rho$. As its expectation is $\frac{\tau}{1-\rho}$, we compared P-EASGD with τ and ρ to vanilla EASGD with $\frac{\tau}{1-\rho}$. As shown in Figure 5.16, the result suggests that P-EASGD with $\tau = 40, \rho = 0.75$ converged faster than vanilla EASGD with $\tau = 160$. As mentioned in Section 2, increasing τ leads to slow convergence as it requires more iterations. We, therefore, conclude that reducing communication through pruning provides better training efficiency compared to having a larger τ. However, in such a case where the size of the parameters to be exchanged is not large (*e.g.,* ResNet-56 for the CIFAR-10 experiment), the effect of pruning diminishes as the time spent for communication becomes less critical.

We emphasize that the traffic burden is the decisive factors to limit the scalability of the proposed framework, since it tends to rise with increase in the number of worker nodes, especially, in the commodity network. As shown in Figs. 5.10 and 5.11, we demonstrated the fact that the convergence worsened despite of allocating of significant amount of resources. P-EASGD suppressed the rate of traffic increment efficiently, while the network traffic increased near-

linearly in a vanilla EASGD setting as illustrated in Figure 5.8. Consequently, the commodity cluster can obtain the additional scalability by P-EASGD.

5.9 Summary

In this chapter, we suggest DeepSpark, a deep learning framework with cost-effective communication for commodity Spark clusters. We demonstrated the effectiveness of the proposed framework using experimental results. The proposed P-EASGD algorithm reduced communication costs up to by 67% with little or no loss. Furthermore, it also suppressed the traffic growth by 35% for deploying 4 times more nodes, while the EASGD setting suffered from the burden over twice. Consequently, we expect the framework to assist other researchers who want to train their own models on commodity Spark clusters.

Interestingly, in the P-EASGD training of ResNet-56 and AlexNet, P-EASGD showed the different convergence with $\alpha = 0.1$. We conjectured that the high pruning rate causes the overfitting of w_i, as can be seen in Figures 5.14 and 5.15, which show a comparison of convergence between P-EASGD and EASGD for the different value of α. Currently, we explored α to compensate those phenomena, manually. Suggesting a theoretical guideline to determine α, ρ, and i_p adaptively will be considered in our future work.

Chapter 6

Conclusion

Functional non-coding RNAs are not themselves encoded as proteins, however they play an important role in regulating each gene expression through interaction with other molecules. There are many methodologies for studying these non-coding RNAs. Among them, we focused on the three kinds of large-scale computational based approaches. First, we built the database for interacting between viral miRNA and host target mRNA. Second, we explored and comparing the structures of RNA molecules from a microscopic point of view. Additionally, as a deep learning based RNA analysis methodology emerged in recent years, we proposed the distributed deep learning framework for supporting those approaches. This chapter summarizes the content and concludes with the effects that may improved or be expected in the future.

- In chapter 2, we built the vHoT database for searching the interaction between viral microRNAs and host genome. Some viruses have been reported to transcribe microRNAs, implying complex relationships between the host and the pathogen at the post-transcriptional level through microRNAs in virus-infected cells. Although many computational algo-

rithms have been developed for microRNA target prediction, few have been designed exclusively to find cellular or viral mRNA targets of viral microRNAs in a user-friendly manner. To address this, we introduce vHoT database for predicting interspecies interactions between viral microRNA and host genomes. Our database supports target prediction of viral microRNAs from human, mouse, rat, rhesus monkey, cow, and virus genomes.

- In chapter 3, we proposed an online server for rapid and robust analysis of large collections of profiles obtained from high-throughput CE experiments. Recent generations of high-throughput DNA and RNA structure mapping studies give hundreds of CE profiles each containing hundreds of bands. To isolate signals from such a large collection of profiles, we previously developed a signal-processing pipeline that consists of multiple stages including preprocessing, alignment, annotation, and band quantification. HiTRACE-Web now enables users to follow the whole pipeline through a user-friendly, integrative, and interactive web environment. It is our hope that HiTRACE-Web can contribute to large-scale structure inference studies based on chemical probing and CE separation by providing an effective and easily accessible data analysis framework.

- In chapter 4, we described comparative analysis of RNA structures which can be useful to infer functions of novel RNAs. In such analysis, it is critical to have a means to measure distances between different RNA molecules, since pairwise distance is often the most fundamental information many data-analysis methods relies on. In this work, we applied a kernel-based approach to measuring distances between RNA structures.

In this methodology, we first represent each RNA structure under study in an intermediate representation. The distance between two RNA structures then becomes the distance between their intermediate representations. To measure their distance, we use a special type of the Mercer kernel, effectively bypassing difficulties in handling RNAs of different lengths and structures. Using real RNA test data, we carry out data-mining tasks such as clustering to test the effectiveness of our approach. We also compare the proposed measure with the conventional edit distance metric for RNA structure comparison in terms of the fidelity to retrieving the labels of the test data.

- In chapter 5, a deep learning framework with cost-effective communication for commodity Spark clusters is suggested to deal with large scale RNA sequences. We also showed its The effectiveness of the proposed framework was shown by the experimental results. The proposed algorithm reduced communication costs by pruning the parameters to be exchanged, successfully. Furthermore, it also suppressed the traffic growth by 35% for deploying even 4 times more nodes, while the vanilla EASGD setting suffered from the burden over twice. Consequently, we expect the framework to assist other researchers who want to train their own models on commodity Spark clusters.

6.1 Future work

The combination of deep learning and RNA sequence analysis shows many possibilities. In large scale analysis RNA sequence analysis, because the preprocessing pipeline must be considered, combining Spark-like distributed pro-

cessing platform will provide improved performance and availability. We will implement the specified model for large scale RNA sequence model instead of general deep learning model, such as AlexNet.

There are also challenges to large-scale distributed optimization for deep models. For the proposed algorithms, quantitative and mathematical analysis are needed to show how the hyperparameter ρ affects learning convergence. Since we expect that it will improve performance and utilization by considering various network topologies presented for high performance computing (HPC) network rather than simple Ethernet, we will explore distributed optimization algorithms for the HPC systems with considering the cluster topologies.

Bibliography

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow. org*, 2015.

[2] Mirela Andronescu, Vera Bereg, Holger H Hoos, and Anne Condon. Rna strand: the rna secondary structure and statistical analysis database. *BMC bioinformatics*, 9(1):340, 2008.

[3] Sharon Aviran, Julius B Lucks, and Lior Pachter. Rna structure characterization from chemical mapping experiments. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 1743–1750. IEEE, 2011.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] David P Bartel. Microrнas: genomics, biogenesis, mechanism, and function. *Cell*, 116(2):281–297, 2004.

[6] Conrad Bessant, Ian Shadforth, and Darren Oakley. *Building Bioinfor-*

matics Solutions: With Perl, R and MySQL. Oxford University Press, Inc., New York, NY, USA, 2009.

[7] Doron Betel, Manda Wilson, Aaron Gabow, Debora S Marks, and Chris Sander. The microrna. org resource: targets and expression. *Nucleic acids research*, 36(suppl_1):D149–D153, 2008.

[8] JP Bida and Rijhu Das. Squaring theory with practice in rna design. *Current opinion in structural biology*, 22(4):457–466, 2012.

[9] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[10] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[11] Tim Bunce and Alligator Descartes. *Programming the Perl DBI: Database programming with Perl*. " O'Reilly Media, Inc.", 2000.

[12] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[13] Chang-Zheng Chen, Ling Li, Harvey F Lodish, and David P Bartel. Microrna modulate hematopoietic lineage differentiation. *Science*, 303(5654):83–86, 2004.

[14] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[15] Trishul M Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, volume 14, pages 571–582, 2014.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[17] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632, 2002.

[18] Pablo Cordero, Wipapat Kladwang, Christopher C VanLang, and Rhiju Das. Quantitative dimethyl sulfate mapping for automated rna secondary structure inference. *Biochemistry*, 51(36):7037–7039, 2012.

[19] Pablo Cordero, Julius B Lucks, and Rhiju Das. An rna mapping database for curating rna structure mapping experiments. *Bioinformatics*, 28(22):3006–3008, 2012.

[20] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[21] Rhiju Das, John Karanicolas, and David Baker. Atomic accuracy in predicting and designing noncanonical rna structure. *Nature methods*, 7(4):291–294, 2010.

[22] Rhiju Das, Madhuri Kudaravalli, Magdalena Jonikas, Alain Laederach, Robert Fong, Jason P Schwans, David Baker, Joseph A Piccirilli, Russ B Altman, and Daniel Herschlag. Structural inference of native and partially folded rna by high-throughput contact mapping. *Proceedings of the National Academy of Sciences*, 105(11):4144–4149, 2008.

[23] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *NIPS*, pages 1223–1231, 2012.

[24] Katherine E Deigan, Tian W Li, David H Mathews, and Kevin M Weeks. Accurate shape-directed rna structure determination. *Proceedings of the National Academy of Sciences*, 106(1):97–102, 2009.

[25] Sean R Eddy. Non–coding rna genes and the modern rna world. *Nature Reviews Genetics*, 2(12):919–929, 2001.

[26] Naama Elefant, Amnon Berger, Harel Shein, Matan Hofree, Hanah Margalit, and Yael Altuvia. Reptar: a database of predicted cellular targets of host and viral mirnas. *Nucleic acids research*, 39(suppl_1):D188–D194, 2010.

[27] Walter Fontana, Danielle AM Konings, Peter F Stadler, and Peter Schuster. Statistics of rna secondary structures. *Biopolymers*, 33(9):1389–1404, 1993.

[28] Eva Gottwein, Neelanjan Mukherjee, Christoph Sachse, Corina Frenzel, William H Majoros, Jen-Tsan A Chi, Ravi Braich, Muthiah Manoharan, Jürgen Soutschek, Uwe Ohler, et al. A viral microrna functions as an ortholog of cellular mir-155. *Nature*, 450(7172):1096, 2007.

[29] Finn Grey, Rebecca Tirabassi, Heather Meyers, Guanming Wu, Shannon McWeeney, Lauren Hook, and Jay A Nelson. A viral microrna down-regulates multiple cell cycle genes through mrna 5′ utrs. *PLoS pathogens*, 6(6):e1000967, 2010.

[30] Sam Griffiths-Jones. The microrna registry. *Nucleic acids research*, 32(suppl_1):D109–D111, 2004.

[31] Sam Griffiths-Jones. mirbase: the microrna sequence database. *MicroRNA Protocols*, pages 129–138, 2006.

[32] Sam Griffiths-Jones, Russell J Grocock, Stijn Van Dongen, Alex Bateman, and Anton J Enright. mirbase: microrna sequences, targets and gene nomenclature. *Nucleic acids research*, 34(suppl_1):D140–D144, 2006.

[33] Sam Griffiths-Jones, Harpreet Kaur Saini, Stijn van Dongen, and Anton J Enright. mirbase: tools for microrna genomics. *Nucleic acids research*, 36(suppl_1):D154–D158, 2007.

[34] Dominic Grün, Yi-Lu Wang, David Langenberger, Kristin C Gunsalus, and Nikolaus Rajewsky. microrna target predictions across seven drosophila species and comparison to mammalian targets. *PLoS computational biology*, 1(1):e13, 2005.

[35] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[37] Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994.

[38] Paul Wei-Che Hsu, Li-Zen Lin, Sheng-Da Hsu, Justin Bo-Kai Hsu, and Hsien-Da Huang. Vita: prediction of host micrornas targets on viruses. *Nucleic acids research*, 35(suppl_1):D381–D385, 2006.

[39] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[40] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics.

[41] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[42] Shuai Jiang, Hong-Wei Zhang, Ming-Hua Lu, Xiao-Hong He, Yong Li, Hua Gu, Mo-Fang Liu, and En-Duo Wang. Microrna-155 functions as

an oncomir in breast cancer by targeting the suppressor of cytokine signaling 1 gene. *Cancer research*, 70(8):3119–3127, 2010.

[43] Bino John, Anton J Enright, Alexei Aravin, Thomas Tuschl, Chris Sander, and Debora S Marks. Human microrna targets. *PLoS biology*, 2(11):e363, 2004.

[44] Fethullah Karabiber, Jennifer L McGinnis, Oleg V Favorov, and Kevin M Weeks. Qushape: rapid, accurate, and best-practices quantification of nucleic acid probing information, resolved by capillary electrophoresis. *Rna*, 19(1):63–73, 2013.

[45] Michael Kertesz, Nicola Iovino, Ulrich Unnerstall, Ulrike Gaul, and Eran Segal. The role of site accessibility in microrna target recognition. *Nature genetics*, 39(10):1278, 2007.

[46] Jinkyu Kim, Seunghak Yu, Byonghyo Shim, Hanjoo Kim, Hyeyoung Min, Eui-Young Chung, Rhiju Das, and Sungroh Yoon. A robust peak detection method for rna structure inference by high-throughput contact mapping. *Bioinformatics*, 25(9):1137–1144, 2009.

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

[48] Marianthi Kiriakidou, Peter T Nelson, Andrei Kouranov, Petko Fitziev, Costas Bouyioukos, Zissimos Mourelatos, and Artemis Hatzigeorgiou. A combined computational-experimental approach predicts human microrna targets. *Genes & development*, 18(10):1165–1178, 2004.

[49] Wipapat Kladwang, Pablo Cordero, and Rhiju Das. A mutate-and-map

strategy accurately infers the base pairs of a 35-nucleotide model rna. *Rna*, 17(3):522–534, 2011.

[50] Wipapat Kladwang and Rhiju Das. A mutate-and-map strategy for inferring base pairs in structured nucleic acids: proof of concept on a dna/rna helix. *Biochemistry*, 49(35):7414–7416, 2010.

[51] Wipapat Kladwang, Christopher C VanLang, Pablo Cordero, and Rhiju Das. A two-dimensional mutate-and-map strategy for non-coding rna structure. *Nature chemistry*, 3(12):954–962, 2011.

[52] Wipapat Kladwang, Christopher C VanLang, Pablo Cordero, and Rhiju Das. Understanding the errors of shape-directed rna structure modeling. *Biochemistry*, 50(37):8049–8056, 2011.

[53] Ana Kozomara and Sam Griffiths-jones. mirbase: integrating microrna annotation and deep-sequencing data. In *Nucleic Acids Res.* Citeseer, 2011.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[55] Alessandro Laganà, Stefano Forte, Francesco Russo, Rosalba Giugno, Alfredo Pulvirenti, and Alfredo Ferro. Prediction of human targets for viral-encoded micrornas by thermodynamics and empirical constraints. *Journal of RNAi and gene silencing: an international journal of RNA and gene targeting research*, 6(1):379, 2010.

[56] Mariana Lagos-Quintana, Reinhard Rauhut, Winfried Lendeckel, and

Thomas Tuschl. Identification of novel genes coding for small expressed rnas. *Science*, 294(5543):853–858, 2001.

[57] Nelson C Lau, Lee P Lim, Earl G Weinstein, and David P Bartel. An abundant class of tiny rnas with probable regulatory roles in caenorhabditis elegans. *Science*, 294(5543):858–862, 2001.

[58] Byunghan Lee, Junghwan Baek, Seunghyun Park, and Sungroh Yoon. deeptarget: End-to-end learning framework for microrna target prediction using deep recurrent neural networks. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '16, pages 434–442, New York, NY, USA, 2016. ACM.

[59] Rosalind C Lee and Victor Ambros. An extensive class of small rnas in caenorhabditis elegans. *Science*, 294(5543):862–864, 2001.

[60] Rosalind C Lee, Rhonda L Feinbaum, and Victor Ambros. The c. elegans heterochronic gene lin-4 encodes small rnas with antisense complementarity to lin-14. *Cell*, 75(5):843–854, 1993.

[61] Benjamin P Lewis, Christopher B Burge, and David P Bartel. Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microrna targets. *cell*, 120(1):15–20, 2005.

[62] Benjamin P Lewis, I-hung Shih, Matthew W Jones-Rhoades, David P Bartel, and Christopher B Burge. Prediction of mammalian microrna targets. *Cell*, 115(7):787–798, 2003.

[63] Min Li, Jian Tan, Yandong Wang, Li Zhang, and Valentina Salapura. Sparkbench: a comprehensive benchmarking suite for in memory data

analytic platform spark. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 53. ACM, 2015.

[64] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598, 2014.

[65] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *NIPS*, pages 2719–2727, 2015.

[66] Changmei Liu, Zhao-Qian Teng, Nicholas J Santistevan, Keith E Szulwach, Weixiang Guo, Peng Jin, and Xinyu Zhao. Epigenetic regulation of mir-184 by mbd1 governs neural stem cell proliferation and differentiation. *Cell stem cell*, 6(5):433–444, 2010.

[67] John A Luckey, Howard Drossman, Anthony J Kostichka, David A Mead, Jonathan D'Cunha, Tracy B Norris, and Lloyd M Smith. High speed dna sequencing by capillary electrophoresis. *Nucleic acids research*, 18(15):4417–4421, 1990.

[68] Manolis Maragkakis, Panagiotis Alexiou, Giorgio L Papadopoulos, Martin Reczko, Theodore Dalamagas, George Giannopoulos, George Goumas, Evangelos Koukis, Kornilios Kourtis, Victor A Simossis, et al. Accurate microrna target prediction correlates with protein repression levels. *BMC bioinformatics*, 10(1):295, 2009.

[69] John S Mattick and Igor V Makunin. Non-coding rna. *Human molecular genetics*, 15(suppl_1):R17–R29, 2006.

[70] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807*, 2015.

[71] Edward J Merino, Kevin A Wilkinson, Jennifer L Coughlan, and Kevin M Weeks. Rna structure analysis at single nucleotide resolution by selective 2 '-hydroxyl acylation and primer extension (shape). *Journal of the American Chemical Society*, 127(12):4223–4231, 2005.

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

[73] Kevin C Miranda, Tien Huynh, Yvonne Tay, Yen-Sin Ang, Wai-Leong Tam, Andrew M Thomson, Bing Lim, and Isidore Rigoutsos. A pattern-based method for the identification of microrna binding sites and their corresponding heteroduplexes. *Cell*, 126(6):1203–1217, 2006.

[74] Somdeb Mitra, Inna V Shcherbakova, Russ B Altman, Michael Brenowitz, and Alain Laederach. High-throughput single-nucleotide structural mapping by capillary automated footprinting analysis. *Nucleic acids research*, 36(11):e63–e63, 2008.

[75] Philipp Moritz, Robert Nishihara, Ion Stoica, and Michael I Jordan. Sparknet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015.

[76] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[77] Pekka Pääkkönen and Daniel Pakkala. Reference architecture and classification of technologies, products and services for big data systems. *Big Data Research*, 2(4):166–186, 2015.

[78] Rahul Palamuttam, Renato Marroquín Mogrovejo, Chris Mattmann, Brian Wilson, Kim Whitehall, Rishi Verma, Lewis McGibbney, and Paul Ramirez. Scispark: Applying in-memory distributed computing to weather event detection and tracking. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 2020–2026. IEEE, 2015.

[79] Phillip S Pang, Menashe Elazar, Edward A Pham, and Jeffrey S Glenn. Simplified rna secondary structure mapping by automation of shape data analysis. *Nucleic acids research*, 39(22):e151–e151, 2011.

[80] Seunghyun Park, Seonwoo Min, Hyunsoo Choi, and Sungroh Yoon. Deep recurrent neural network-based identification of precursor micrornas. In *NIPS*, 2017.

[81] Mateusz Pawlik and Nikolaus Augsten. Rted: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[82] Sébastien Pfeffer, Mihaela Zavolan, Friedrich A Grässer, Minchen Chien, James J Russo, Jingyue Ju, Bino John, Anton J Enright, Debora Marks, Chris Sander, et al. Identification of virus-encoded microrna. *Science*, 304(5671):734–736, 2004.

[83] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[84] Marc Rehmsmeier, Peter Steffen, Matthias Höchsmann, and Robert Giegerich. Fast and effective prediction of microrna/target duplexes. *Rna*, 10(10):1507–1517, 2004.

[85] Brenda J Reinhart, Frank J Slack, Michael Basson, Amy E Pasquinelli, Jill C Bettinger, Ann E Rougvie, H Robert Horvitz, and Gary Ruvkun. The 21-nucleotide let-7 rna regulates developmental timing in caenorhabditis elegans. *Nature*, 403(6772):901–906, 2000.

[86] Peter Sarnow, Catherine L Jopling, Kara L Norman, Sylvia Schütz, and Karen A Wehner. Micrornas: expression, avoidance and subversion by vertebrate viruses. *Nature reviews. Microbiology*, 4(9):651, 2006.

[87] Alexander Sczyrba, Jan Kruger, Henning Mersch, Stefan Kurtz, and Robert Giegerich. Rna-related tools on the bielefeld bioinformatics server. *Nucleic acids research*, 31(13):3767–3770, 2003.

[88] Matthias Selbach, Björn Schwanhäusser, Nadine Thierfelder, Zhuo Fang, Raya Khanin, and Nikolaus Rajewsky. Widespread changes in protein synthesis induced by micrornas. *nature*, 455(7209):58, 2008.

[89] Praveen Sethupathy, Benoit Corda, and Artemis G Hatzigeorgiou. Tarbase: A comprehensive database of experimentally supported animal microrna targets. *Rna*, 12(2):192–197, 2006.

[90] Bruce A Shapiro. An algorithm for comparing multiple rna secondary structures. *Bioinformatics*, 4(3):387–393, 1988.

[91] Bruce A Shapiro and Kaizhong Zhang. Comparing multiple rna secondary structures using tree comparisons. *Bioinformatics*, 6(4):309–318, 1990.

[92] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *MSST*, pages 1–10. IEEE, 2010.

[93] Pilar Tijerina, Sabine Mohr, and Rick Russell. Dms footprinting of structured rnas and rna–protein complexes. *Nature protocols*, 2(10):2608–2623, 2007.

[94] Jennifer Lin Umbach, Martha F Kramer, Igor Jurak, Heather W Karnowski, Donald M Coen, and Bryan R Cullen. Microrna expressed by herpes simplex virus 1 during latent infection regulate viral mrnas. *Nature*, 454(7205):780, 2008.

[95] Suzy M Vasa, Nicolas Guex, Kevin A Wilkinson, Kevin M Weeks, and Morgan C Giddings. Shapefinder: a software system for high-throughput quantitative analysis of nucleic acid reactivity information resolved by capillary electrophoresis. *Rna*, 14(10):1979–1990, 2008.

[96] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, page 5. ACM, 2013.

[97] Robert Walczak, Eric Westhof, Philippe Carbon, and Alain Krol. A novel rna structural motif in the selenocysteine insertion element of eukaryotic selenoprotein mrnas. *Rna*, 2(4):367–379, 1996.

[98] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings*

of the 30th International Conference on Machine Learning (ICML-13), pages 1058–1066, 2013.

[99] Wenjuan Wang, Taoying Liu, Dixin Tang, Hong Liu, Wei Li, and Rubao Lee. Sparkarray: An array-based scientific data management system built on apache spark. In *Networking, Architecture and Storage (NAS), 2016 IEEE International Conference on*, pages 1–10. IEEE, 2016.

[100] Joseph M Watts, Kristen K Dang, Robert J Gorelick, Christopher W Leonard, Julian W Bess Jr, Ronald Swanstrom, Christina L Burch, and Kevin M Weeks. Architecture and secondary structure of an entire hiv-1 rna genome. *Nature*, 460(7256):711–716, 2009.

[101] Kevin M Weeks. Advances in rna structure analysis by chemical probing. *Current opinion in structural biology*, 20(3):295–304, 2010.

[102] Robert Weinberger. *Practical capillary electrophoresis*. Academic Press, 2000.

[103] Marek S Wiewiórka, Antonio Messina, Alicja Pacholewska, Sergio Maffioletti, Piotr Gawrysiak, and Michał J Okoniewski. Sparkseq: fast, scalable, cloud-ready tool for the interactive genomic data analysis with nucleotide precision. *Bioinformatics*, page btu343, 2014.

[104] Kevin A Wilkinson, Robert J Gorelick, Suzy M Vasa, Nicolas Guex, Alan Rein, David H Mathews, Morgan C Giddings, and Kevin M Weeks. High-throughput shape analysis reveals structures in hiv-1 genomic rna strongly conserved across distinct biological states. *PLoS biology*, 6(4):e96, 2008.

[105] Adam T Woolley and Richard A Mathies. Ultra-high-speed dna sequencing using capillary electrophoresis chips. *Analytical chemistry*, 67(20):3676–3680, 1995.

[106] Feifei Xiao, Zhixiang Zuo, Guoshuai Cai, Shuli Kang, Xiaolian Gao, and Tongbin Li. mirecords: an integrated resource for microrna–target interactions. *Nucleic acids research*, 37(suppl_1):D105–D110, 2008.

[107] Sungroh Yoon, Jinkyu Kim, Justine Hum, Hanjoo Kim, Seunghyun Park, Wipapat Kladwang, and Rhiju Das. Hitrace: high-throughput robust analysis for capillary electrophoresis. *Bioinformatics*, 27(13):1798–1805, 2011.

[108] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[109] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, volume 10, page 10, 2010.

[110] Jeremy D Zawodny and Derek J Balling. *High Performance MySQL: Optimization, Backups, Replication, Load Balancing & More*. " O'Reilly Media, Inc.", 2004.

[111] Hao Zhang, Zhiting Hu, Jinliang Wei, Pengtao Xie, Gunhee Kim,

Qirong Ho, and Eric Xing. Poseidon: A system architecture for efficient gpu-based deep learning on multiple machines. *arXiv preprint arXiv:1512.06216*, 2015.

[112] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *NIPS*, pages 685–693, 2015.

[113] Michael Zuker and David Sankoff. Rna secondary structures and their prediction. *Bulletin of mathematical biology*, 46(4):591–621, 1984.

초 록

2000년대에 들어와서 리보핵산 (RNA)의 연구는 새로운 전기를 맞이하게 되었다. 인간 게놈 프로젝트등의 결과로 알려진 바 실제로 단백질로 작용 번역되는 유전자의 비율은 수 %에 불과했고, 대부분의 non-coding gene 들은 직접 여러가지 생체 반응이나 유전자 발현에 참여하고 있다. miRNA로 대표되는 이러한 기능성 비발현 RNA 들의 연구는 빅데이터 시대를 맞이한 컴퓨터 정보기술의 발전에 힘입어 더욱 가속화되고 있다.

그 동안 축적되어온 RNA의 염기서열과 구조, 그리고 그들이 이루는 상호작용에 대한 정보량은 데이터의 규모에서 보았을 때, 대규모 처리를 위한 여러가지 기법들을 적용하여 그 효율을 더욱 향상시킬 수 있음을 기대할 수 있다. 본 학위논문에서는 RNA의 상호작용 예측과 구조비교와 분석에 관한 방법론, 그리고 마지막으로 최근들어 RNA 분석 접근 방법 중 급부상하고 있는 딥러닝 기반의 접근 방식을 분산처리하는 프레임워크를 제안하였다.

첫 번째로 다룬 것은 miRNA와 mRNA의 상호작용을 예측하는 것이다. 몇몇 종에 대한 자가 miRNA-mRNA 간의 상호작용에 대한 연구는 많이 이루어져 있지만, 바이러스와 그 숙주에 관한 상호작용은 대규모로 탐구된 사례가 없었다. 우리는 바이러스의 miRNA와 숙주의 mRNA 간의 상호작용을 여러가지 알고리즘을 이용해 예측하고 미리 계산된 결과를 대규모 데이터베이스로 구축하고 빠른 검색과 비교를 가능하게 하였다. 두 번째는 RNA의 구조에 관한 문제이다. 우리는 고비용을 요구하는 물리적인 방법 대신에 화학적인 방법을 통해 얻어진 RNA 구조에 관한 정보를 정량화하여 분석할 수 있는 온라인 도구를 제안하여 RNA의 구조를 예측할 수 있도록 하였다. 더불어 얻어진 구조정보를 적절하게 비교하는 비교 방법들을 비교하고 트리기반의 거리 측정 알고리즘을 이용하여 실제로 유효한 분류 결과를 얻을 수 있음을 보였다. 마지막으로, 현재 RNA 염기서열 비교나

식별에서 좋은 결과를 보여주고 있는 딥러닝 기반 접근을 위한 분산 플랫폼을 제안하였다. Apache Spark 기반의 저비용 분산 시스템에 데이터 병렬화 기반의 딥러닝 학습을 수행할 수 있는 알고리즘과 구현체를 제안하여 실제로 성능향상을 보이고 그 이용 가능성을 제시하였다.

주요어: RNA 염기서열, 생물정보학, 대규모 데이터 처리, 데이터베이스, 기계 학습, 딥러닝, 병렬처리, 분산 컴퓨팅

학번: 2013-30232