공학박사 학위논문

Denoising and Interaction Learning of Biological Data

생체 자료 오류 정정 및 관계 학습

2018년 2월

서울대학교 대학원

전기·컴퓨터공학부

이 병 한

Denoising and Interaction Learning of Biological Data

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함

2017년 12월

서울대학교 대학원

전기·컴퓨터공학부

이 병 한

이병한의 공학박사 학위논문을 인준함

2017년 12월

위 원 장 강 신 (인)

부위원장 윤 성 로 (인)

위 원 천 종 식 (인)

위 원 박 대 희 (인)

위 원 문 태 섭 (인)

Abstract

Since the Human Genome Project was completed, enormous biological data have been accumulated as an attempt to understand the biological mechanisms of human. However, errors induced during the sequencing procedures and unrevealed inherent features of biological data for inferring their interactions arouse the necessity of large-scale data-driven applications. In this regard, this dissertation exploits the recent advances in machine learning and artificial intelligence techniques that have shown their success in time series sequence learning, including natural language processing and neural machine translation, to improve the reliability and computational performance of investigating biological data.

This dissertation discusses three issues in sequence analysis and proposes methodologies to overcome them. First, to alleviate the error-prone nature of sequence reads from next-generation sequencing (NGS), we present an information theoretic approach for correcting sequence errors from various sequencers. Next, we show a generalized multi-graphics processing units (GPUs) accelerated sequence denoiser to address the computational challenges of denoising high-throughput sequences. Finally, we describe an end-to-end machine learning framework for robust sequence (*e.g.*, miRNA) target prediction to boost the sensitivity without the laborious manual feature extraction procedure. In summary, this dissertation proposes a set of methodologies on the basis of machine learning algorithms to handle biological sequences that can boost the reliability of downstream analysis.

Keywords: machine learning, deep learning, end-to-end learning, parallelization, sequence error, sequence interaction, time series, miRNA target

Student Number: 2012-23229

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A new generation of high-throughput, low-cost sequencing technologies, referred to as next-generation sequencing (NGS) technologies [1], is reshaping biomedical research, including large-scale comparative and evolutionary studies [2, 3, 4]. Compared with automated Sanger sequencing, NGS platforms produce significantly shorter reads in large quantities, posing various new computational challenges [5].

There are several DNA sequencing methodologies that use NGS [6, 7] including whole genome sequencing (WGS), chromatin immunoprecipitation (ChIP) sequencing, and targeted sequencing. WGS is used to analyze the genome of an organism to capture all variants and identify potential causative variants; it is also used for *de novo* genome assembly. ChIP sequencing identifies genome-wide DNA binding sites for transcription factors and other proteins. Targeted sequencing (*e.g.*, exome sequencing and amplicon sequencing), the focus of this paper, is a cost-effective method that enables researchers to focus on investigating areas of interest that are likely to be involved in a particular phenotype. According to previous studies [8, 9], targeted sequencing often results in the complete coverage of exons of disease-related genes, while alternative methods result in approximately 90–95% coverage. Hence, in clinical settings, researchers tend to rely on targeted sequencing for diagnostic

evaluations.

To detect sequences based on fluorescent labels at the molecular level, NGS technologies normally rely on imaging systems requiring templates that are amplified by emulsion polymerase chain reaction (PCR) or solid-phase amplification [1]. These amplification and imaging processes can generate erroneous reads, the origin of which can be traced to the incorrect determination of homopolymer lengths, the erroneous insertion/deletion/substitution of nucleotide bases, and PCR chimeras [6]. Substitution errors dominate for many platforms, including Illumina, while homopolymer errors, manifested as insertions and deletions (indels), are also abundant for 454 pyrosequencing and Ion Torrent.

Erroneous reads must be properly handled because they complicate downstream analyses (*e.g.*, variant calling and genome assembly), often lowering the quality of the whole analysis pipeline [7]. Soft clipping, in which 3'-ends of a read are trimmed based on the quality scores of individual bases, may be the simplest approach, but it results in a loss of information [10]. More sophisticated methods focus on detecting and correcting errors in sequence data [11, 12, 13, 14, 15, 16, 17, 18, 19, 20].

As summarized in recent reviews [10, 21], current error-correction methods for NGS data can be categorized as follows: k-mer (*i.e.*, oligonucleotides of length k) frequency/spectrum-based, multiple sequence alignment (MSA)-based, and statistical error model-based methods. While the above methods often exhibit good performance for various platforms, they also have several limitations. First, k-mer-based schemes tend to be ineligible when the coverage is expected to vary over the queried sequences, as in transcriptomics, metagenomics, heterogeneous cell samples, or pre-amplified libraries [21]. Second, MSA-based methods, which do not suffer from the above issue related to non-uniform coverage, often require the application of heuristic and sophisticated consensus decision rules for the aligned columns, and such rules may

be sensitive to specific applications or sequencing platforms. Third, statistical error model-based methods typically use computationally expensive schemes (*i.e.*, expectation-maximization) owing to additional stochastic modeling assumptions for the underlying DNA sequences. Moreover, little attention is given to the validity and accuracy of such modeling assumptions, let alone to theoretical analysis of whether near optimum or sound error-correction performance is attained. Finally, many existing schemes applying the three methods often return only representative (consensus) denoised sequences created by merging input sequences; hence, the number of sequences is often not preserved after denoising. In some applications, this may result in inconsistencies in downstream analyses.

To address the limitations of existing approaches, this dissertation proposes an information theoretic approach for correcting sequence errors from various sequencers. Our methodology exploits an algorithm called Discrete Universal DEnoiser (DUDE) [22], which was developed for a general setting of reconstructing sequences with finite-valued components (source symbols) corrupted by a noise mechanism that corrupts each source symbol independently and statistically identically. The contents of this chapter are based on the following research:

- Byunghan Lee, Taesup Moon, Sungroh Yoon, and Tsachy Weissman, "DUDE-Seq: Fast, flexible, and robust denoising for targeted amplicon sequencing," *PLOS ONE*, 12(7):e0181463, July 2017.

In addition to the theoretical perspective, existing approaches [23, 24, 25, 26, 27] require a long time, typically taking days to process only a part of the collected amplicons. Clearly, this slow rate bottlenecks the entire analysis pipeline, apart from hindering efforts to improve robustness. In other disciplines that require heavy computation, the use of general-purpose computing on graphics processing units (GPGPUs) has become increasingly popular. GPGPU-based approaches utilize GPUs for accelerating a wide range of

applications other than computation for computer graphics [28, 29]. In bioinformatics research, the scientists have achieved large performance benefits on compute-intensive sequence analysis tasks (*e.g.*, sequence classification and comparison [30, 31]) by using GPUs. Recent advances in GPU technology allow us to connect multiple GPUs through the system bus or dedicated communication links to achieve even higher parallelization performance.

To address the computational challenge of denoising high-throughput sequences, this dissertation proposes a generalized multi-GPU accelerated sequence denoiser. Our methodology exploits a data-level parallelism underlying a generalized clustering-based denoiser for targeted sequencing and is implemented on top of multiple GPUs and CPUs on cluster systems. The contents of this chapter are based on the following research:

- Byunghan Lee, Hyeyoung Min, and Sungroh Yoon, "MUGAN: Multi-GPU accelerated AmpliconNoise server for rapid microbial diversity assessment," *Bioinformatics*, under revision.

- Byunghan Lee, Joonhong Park, and Sungroh Yoon, "Rapid and Robust Denoising of Pyrosequenced Amplicons for Metagenomics," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, Brussels, Belgium, December 2012.

Given the preprocessed biological sequences, researchers are concentrated on investigating interactions between sequences at atomic/molecular levels. Among the various interactions between them, investigating microRNAs (miRNAs), which play a central role in the post-transcriptional regulation of more than 60% of protein coding genes [32] by controlling the expression of target messenger RNAs (mRNAs), is of utmost importance in many disciplines of life science.

Due to the importance, there has been a deluge of computational algorithms proposed to address the miRNA target prediction problem [33, 34, 35].

However, most of the existing approaches suffer from two major limitations, often failing to deliver satisfactory performance in practice. First, the conventional approaches make predictions based on the manually crafted features of known miRNA-mRNA pairs (*e.g.*, the level of sequence complementarity between a miRNA sequence and the sequence of a binding site in its target mRNA). Manual feature extraction is time-consuming, laborious, and error-prone, giving no guarantee for generality. Second, existing tools often fail to effectively filter out false positives (*i.e.*, bogus miRNA-mRNA pairs that do not actually interact *in vivo*), producing an unacceptable low level of specificity. This challenge originates from the fact that there are only four types of letters (`A`, `C`, `G`, and `U`) in RNA sequences and that the length of a typical miRNA sequence is short (about 22 nucleotides), producing a high chance of seeing a random site (in mRNA) whose sequence is complementary to the miRNA.

To boost the sensitivity of miRNA target prediction, a variety of features have been proposed. According to Menor *et al.* [36], as many as 151 kinds of features appear in the literature, which can be broadly grouped into four common types [34]: the degree of Watson-Crick matches of a seed sequence; the degree of sequence conservation across species; Gibbs free energy, which measures the stability of the binding of a miRNA-mRNA pair, and the site accessibility, which measures the hybridization possibility of a pair from their secondary structures.

To address the limitations of existing approaches, this dissertation proposes an end-to-end machine learning framework for robust miRNA target prediction. The performance gap between our methodology and the compared alternatives is substantial (over 25% increase in F-measure), and our methodology delivers a quantum leap in the long-standing challenge of robust miRNA target prediction. The contents of this chapter are based on the following research:

- 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 Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB)*, Seattle, USA, October 2016.

The remainder of the dissertation is organized as follows. Chapter 2 presents an information theoretic approach for correcting sequence errors from various sequencers. In Chaper 3, we show a generalized multi-GPU accelerated sequence denoiser. In Chaper 4, we describe an end-to-end machine learning framework for robust miRNA target prediction.

Chapter 2

Sequence Denoising

As summarized in recent reviews [10, 21], current error-correction methods for NGS data can be categorized as follows: k-mer (*i.e.*, oligonucleotides of length k) frequency/spectrum-based, multiple sequence alignment (MSA)-based, and statistical error model-based methods. The idea behind k-mer-based methods [13, 20, 37, 38, 39, 40] is to create a list of "trusted" k-mers from the input reads and correct untrusted k-mers based on a consensus represented by this spectrum. In addition to the length of the k-mer, coverage (k-mer occurrences) information is important to determine trusted k-mers. Under the assumption that errors are rare and random and that coverage is uniform, for sufficiently large k, it is reasonable to expect that most errors alter k-mers to inexistent ones in a genome. Thus, for high-coverage genome sequences obtained by NGS, we may identify suspicious k-mers and correct them based on a consensus. MSA-based methods [16, 12, 41] work by aligning related sequences according to their similarities and correcting aligned reads, usually based on a consensus in an alignment column, using various techniques. This alignment-based scheme is inherently well-suited for correcting indel errors. Early methods suffered from computational issues, but recent approaches utilize advanced indexing techniques to expedite the alignments. In statistical error model-based methods [42, 43, 44], a statistical model is developed to

capture the sequencing process, including error generation. In this regard, an empirical confusion model is often created from datasets, exploiting the information obtained from, *e.g.*, alignment results, Phred quality scores (a measure of the quality of nucleobases generated by automated DNA sequencing) [45], or other parameters.

In this chapter, as an alternative, we applied an algorithm called Discrete Universal DEnoiser (DUDE) [22] for accurate DNA sequence denoising. DUDE was developed for a general setting of reconstructing sequences with finite-valued components (source symbols) corrupted by a noise mechanism that corrupts each source symbol independently and statistically identically. In the DNA denoising literature, such a noise model is equivalent to the confusion matrix commonly used in statistical error-model-based methods. As demonstrated in the original paper [22], DUDE exhibits rigorous performance guarantee for the following setting; even when no stochastic modeling assumptions are made for the underlying clean source data, only with the assumption of *known* noise mechanism, DUDE is shown to universally attain the optimum denoising performance for *any* source data the data increase. We note that the above setting of DUDE naturally fits the setting for DNA sequence denoising, *i.e.*, it is difficult to establish accurate stochastic models for clean DNA sequences, but it is simple and fairly realistic to assume noise models (*i.e.*, confusion matrices) for sequencing devices based on reference sequences.

The DUDE algorithm, which will be explained in details in the next section, possesses flavors that are somewhat connected to all three representative methods mentioned above, in a single scheme. Specifically, DUDE works with double-sided contexts of a fixed size that are analogous to k-mers. Moreover, like MSA, DUDE applies a denoising decision rule to each noisy symbol based on aggregated information over certain positions in the reads. However, unlike MSA, which makes a decision based on the information collected from the symbols in the same aligned column, DUDE makes a decision using the infor-

mation collected from positions with the same double-sided context. Finally, the denoising decision rule of DUDE utilizes information from the assumed noise model, like in most statistical error model-based methods, but does not assume any stochastic model on the underlying sequence, thus resulting in a computationally efficient method. The method of incorporating the noise model is also simple, making it easy to flexibly apply DUDE to different sequencing platforms by simply changing the confusion matrix model in the algorithm.

With the above unique nature of the DUDE algorithm, we show in our experiments that it outperforms other state-of-the-art schemes, particularly for applications to targeted amplicon sequencing. Specifically, among the applicable areas of targeted amplicon sequencing (*e.g.*, cancer gene, 16S rRNA, plant, and animal sequencing [46]), we used 16S rRNA benchmark datasets obtained with different library preparation methods and DNA polymerases to confirm the robustness of our algorithm across various sequencing preparation methods. Targeted amplicon sequencing datasets often have deeper sequencing coverage than those of WGS or ChIP datasets, which frequently makes conventional k-mer-based techniques often suffer from the amplification bias problem [47]. By contrast, for DUDE-Seq, as the sequencing coverage becomes deeper, context-counting vectors can accumulate more probable contexts, and the robustness of denoising typically improves. We apply two versions of DUDE separately for substitution and homopolymer errors, the two major types of sequencing error. For substitution errors, our approach directly utilizes the original DUDE with appropriate adaptation to DNA sequences and is applicable to reads generated by any sequencing platform. For homopolymer errors, however, we do not apply the original DUDE, which was developed in a framework that does not cover errors of the homopolymer type. To correct homopolymer errors, we therefore adopt a variant of DUDE for general-output channels [48]. Our homopolymer-error correction is applicable to cases in which base-called

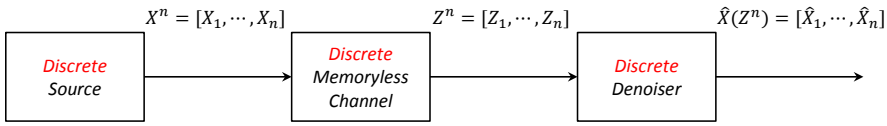


Figure 2.1: The general setting of discrete denoising.

sequences and the underlying flowgram intensities are available (*e.g.*, pyrosequencing and Ion Torrent). For brevity, we refer to both of these DUDE-based approaches as DUDE-Seq, but the correction type will be easily distinguishable by the reader.

2.1 Background

2.1.1 Discrete Universal DEnoiser (DUDE)

In this section, we formally introduce the DUDE algorithm along with its notation and its connection to DNA sequence denoising. Figure 2.1 shows the concrete setting of the discrete denoising problem. We denote the underlying source data as $\{x_i\}$ and assume each component takes values in some finite set $\mathcal{X}$. The resulting noisy version of the source corrupted by a noise mechanism is denoted as $\{Z_i\}$, and its components take values in, again, some finite set $\mathcal{Z}$. As mentioned in the Introduction, DUDE assumes that the noise mechanism injects noises that are independent and statistically identical, and such a mechanism is often referred to as a Discrete Memoryless Channel (DMC) in information theory. The DMC is completely characterized by the channel transition matrix, also known as the confusion matrix, $\mathbf{\Pi} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{Z}|}$, of which the (x, z)-th element, $\Pi(x, z)$, stands for $\Pr(Z_i = z|x_i = x)$, *i.e.*, the conditional probability that the noisy symbol takes value z, given that the original source symbol is x. We denote random variables with uppercase letters and the individual samples of random variables or deterministic symbols with lowercase letters. Thus, the underlying source data, which are treated by DUDE as individual sequences (and not a stochastic process), are denoted by the

lowercase $\{x_i\}$, and the noise-corrupted sequences, *i.e.*, sequences of random variables, are denoted by uppercase $\{Z_i\}$. Furthermore, throughout this paper, we generally denote a sequence (n-tuple) as $a^n = (a_1, \ldots, a_n)$, for example, where a_i^j refers to the subsequence $(a_i, \ldots, a_j)$.

As shown in Figure 2.1, a discrete denoiser observes the entire noisy data Z^n and reconstructs the original data with $\hat{X}^n = (\hat{X}_1(Z^n), \ldots, \hat{X}_n(Z^n))$. The goodness of the reconstruction by a discrete denoiser $\hat{X}^n$ is measured by the average loss,

$$L_{\hat{X}^n}(x^n, Z^n) = \frac{1}{n} \sum_{i=1}^{n} \Lambda(x_i, \hat{X}_i(Z^n)), \tag{2.1}$$

where $\Lambda(x_i, \hat{x}_i)$ is a single-letter loss function that measures the loss incurred by estimating x_i with $\hat{x}_i$ at location i. The loss function can be also represented with a loss matrix $\mathbf{\Lambda} \in \mathbb{R}^{|\mathcal{X}| \times |\hat{\mathcal{X}}|}$.

DUDE in Weissman *et al.* [22] is a two-pass algorithm that has linear complexity with respect to the data size n. During the first pass, given the realization of the noisy sequence z^n, the algorithm collects the statistics vector

$$\mathbf{m}(z^n, l^k, r^k)[a] = \left|\{i : k+1 \le i \le n-k, z_{i-k}^{i+k} = l^k a r^k\}\right|,$$

for all $a \in \mathcal{Z}$, which is the count of the occurrence of the symbol $a \in \mathcal{Z}$ along the noisy sequence z^n that has the *double-sided context* $(l^k, r^k) \in \mathcal{Z}^{2k}$. Note that $\mathbf{m}$ is similar to the counts across the aligned columns for the simple majority voting in MSA-based denoising methods. However, in DUDE, the count is collected regardless of whether the positions in the reads are aligned or not, but considering whether the position has the same context. Additionally, the context length k is analogous to the k-mer length. Once the $\mathbf{m}$ vector is collected, for the second pass, DUDE then applies the rule

$$\hat{X}_i(z^n) = \arg\min_{\hat{x} \in \mathcal{X}} \mathbf{m}^T(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k}) \mathbf{\Pi}^{-1}[\lambda_{\hat{x}} \odot \pi_{z_i}] \tag{2.2}$$

for each $k+1 \leq i \leq n-k$, where π_{z_i} is the z_i-th column of the channel matrix $\mathbf{\Pi}$, and $\lambda_{\hat{x}}$ is the $\hat{x}$-th column of the loss matrix $\mathbf{\Lambda}$. Furthermore, $\odot$ stands for the element-wise product operator for two vectors. The intuitive explanation of Eq. 2.2 is as follows: when we rearrange the right-hand side of Eq. 2.2, we obtain

$$(2.2) \quad = \quad \arg\min_{\hat{x} \in \mathcal{X}} \lambda_{\hat{x}}^T \{\pi_{z_i} \odot \mathbf{\Pi}^{-T} \mathbf{m}^T(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})\}, \tag{2.3}$$

and we can show that $\pi_a \odot \mathbf{\Pi}^{-T} \mathbf{m}^T(z^n, l^k, r^k)$ approximates the empirical count vector of the underlying *clean* symbol at the middle location that resulted in the noisy context $l^k a r^k$. Thus, the denoising rule Eq. 2.2, re-expressed in Eq. 2.3, finds a reconstruction symbol $\hat{x}$ that minimizes the expected loss with respect to the *empirical estimate* (obtained by utilizing the inverse of $\mathbf{\Pi}$) of the count vector of the underlying x_i given the noisy context z_{i-k}^{i+k}. At a high level, DUDE is not a simple majority voting rule based on $\mathbf{m}$; instead, it incorporates the DMC model $\mathbf{\Pi}$ (the confusion matrix) and loss function $\mathbf{\Lambda}$ to obtain a more accurate estimation of the clean source symbol. For more detailed and rigorous arguments on the intuitive description of Eq. 2.2, we refer readers to the original paper [22, Section IV-B].

Note that formula (2.2) assumes $\mathcal{X} = \mathcal{Z} = \hat{\mathcal{X}}$ and $\mathbf{\Pi}$ is invertible for simplicity, but Weissman *et al.* [22] deal with more general cases as well. The form of Eq. 2.2 also shows that DUDE is a sliding window denoiser with window size $2k+1$; *i.e.*, DUDE returns the same denoised symbol at all locations with the same value z_{i-k}^{i+k}. DUDE is guaranteed to attain the optimum performance by the sliding window denoisers with the same window size as the observation length n increases. For more details on the theoretical performance analyses, see Weissman *et al.* [22, Section V].

The original DUDE dealt exclusively with the case of $|\mathcal{X}|$ and $|\mathcal{Z}|$ finite. Dembo and Weissman [48] generalized DUDE to the case of discrete input and general output channels; the noisy outputs do not have to have their values

in some finite set, but can have continuous values as well. As in Weissman *et al.* [22], the memoryless noisy channel model, which is characterized in this case by the set of densities $\{f_x\}_{x\in\mathcal{X}}$, was assumed to be known. As shown in [48, Fig 1], the crux of the arguments is to apply a scalar quantizer $Q(\cdot)$ to each continuous-valued noisy output $\{Y_i\}$ and to derive a virtual DMC, $\mathbf{\Gamma} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{Z}|}$, between the discrete input $\{X_i\}$ and the quantized (hence, discrete) output $\{Z_i\}$. Such $\mathbf{\Gamma}$ can be readily obtained by the knowledge of $\{f_x\}_{x\in\mathcal{X}}$ and evaluating the following integral for each (x, z): $\Gamma(x, z) = \int_{y:Q(y)=z} f_x(y)dy$. Once the virtual DMC is obtained, the rest of the algorithm in Dembo and Weissman [48] proceeds similarly as the original DUDE; specifically, it obtains the statistics vector $\mathbf{m}$ for the quantized noisy outputs $\{Z_i\}$ during the first pass, and then applies a sliding window denoising rule similar to Eq. 2.2, which depends on the statistics vector $\mathbf{m}$, the virtual DMC $\mathbf{\Gamma}$, $\{f_x\}_{x\in\mathcal{X}}$, and the noisy sequence Y^n, during the second pass. A concrete denoising rule can be found in Dembo and Weissman [48, Eqs. (16),(19), and (20)]. In Dembo and Weissman [48], a formal analysis of the generalized DUDE shows that it attains the optimum denoising performance among sliding window denoisers with the same window size, that base their denoising decisions on the original continuous-valued outputs Y^n. We refer readers to the paper for more details. In the next section, we show how we adopt this generalized DUDE in our DUDE-Seq to correct homopolymer errors in DNA sequencing.

2.2 Methods

2.2.1 Substitution Errors

As described in the previous section, the setting of the original DUDE algorithm naturally aligns with the setting of substitution-error correction in DNA sequence denoising. We can set $\mathcal{X} = \mathcal{Z} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, and the loss function as the Hamming loss, namely, $\Lambda(x, \hat{x}) = 0$, if $x = \hat{x}$, and $\Lambda(x, \hat{x}) = 1$, other-

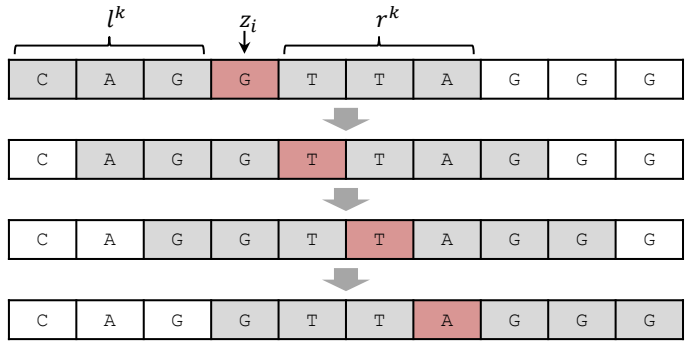


Figure 2.2: A sliding window procedure of the DUDE-Seq with the context size $k = 3$.

wise. Then, the two-pass sliding window procedure of DUDE for collecting the statistics vector $\mathbf{m}$ and the actual denoising can be directly applied as shown in the toy example in Figure 2.2. Before we formally describe our DUDE-Seq for substitution-error correction, however, we need to address some subtle points.

First, the original DUDE in Eq. 2.2 assumes that the DMC matrix $\mathbf{\Pi}$ is known beforehand, but in real DNA sequence denoising, we need to estimate $\mathbf{\Pi}$ for each sequencing device. As described in the Experimental Results section in detail, we performed this estimation following the typical process for obtaining the empirical confusion matrix, *i.e.*, we aligned the predefined reference sequence and its noise-corrupted sequence and then determined the ratio of substitution errors and obtain the estimated $\mathbf{\Pi}$. Second, the original DUDE assumes that the noise mechanism is memoryless, *i.e.*, the error rate does not depend on the location of a base within the sequence. In contrast, for real sequencing devices, the actual error rate, namely, the conditional probability $\Pr(Z_i = z|X_i = x)$ may not always be the same for all location index values i. For example, for Illumina sequencers, the error rate tends to increase towards the ends of reads, as pointed out in Laehnemann *et al.* [21]. In our DUDE-Seq, however, we still treat the substitution error mechanism as a DMC and therefore use the single estimated $\mathbf{\Pi}$ obtained as above, which is essentially the same as that obtained using the *average* error rate matrix. Our experimen-

tal results show that such an approach still yields very competitive denoising results. Thirdly, the optimality of the original DUDE relies on the stationarity of the underlying clean sequence, thus requiring a very large observation sequence length n to obtain a reliable statistics vector $\mathbf{m}$. In contrast, most sequencing devices generate multiple short reads of lengths 100–200. Hence, in DUDE-Seq, we combined all statistics vectors collected from multiple short reads to generate a single statistics vector $\mathbf{m}$ to use in Eq. 2.2.

Addressing the above three points, a formal summary of DUDE-Seq for the substitution errors is given in Algorithm 1. Note that the pseudocode in Algorithm 1 skips those bases whose Phred quality scores are higher than a user-specified threshold and invokes DUDE-Seq only for the bases with low quality scores (lines 10–14). This is in accord with the common practice in sequence preprocessing and is not a specific property of the DUDE-Seq algorithm. Furthermore, for simplicity, we denoted z^n as the entire noisy DNA sequence, and $\mathbf{m}^T(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})$ represents the aggregated statistics vector obtained as described above.

Remarks

1. Incorporating flanking sequences in DUDE-Seq is quite straightforward; we can simply use the one-sided contexts l^{2k} or r^{2k} once DUDE-Seq reaches the flanking regions. In our experiments, however, we did not perform such modification (lines 7–8 of Algorithm 1) since we normally used small k values (around $k = 5$). As demonstrated in our experimental results, the effect of such small flanking regions is not significant on the final denoising results, and we can achieve satisfactory results without considering flanking regions. However, in general, should longer values of k be needed, we can easily modify the algorithm to incorporate one-sided contexts in the flanking regions, and such modification will clearly improve the final denoising result.

Algorithm 1 The DUDE-Seq for substitution errors

Require: Observation z^n, Estimated DMC matrix $\mathbf{\Pi} \in \mathbb{R}^{4\times4}$, Hamming loss $\mathbf{\Lambda} \in \mathbb{R}^{4\times4}$, Context size k, Phred quality score Q^n

Ensure: The denoised sequence $\hat{X}^n$

Define $\mathbf{m}(z^n, l^k, r^k) \in \mathbb{R}^4$ for all $(l^k, r^k) \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^{2k}$.
Initialize $\mathbf{m}(z^n, l^k, r^k)[a] = 0$ for all $(l^k, r^k) \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^{2k}$ and for all $a \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$
for $i \leftarrow k+1, \ldots, n-k$ **do** ▷ First pass
    $\mathbf{m}(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})[z_i] = \mathbf{m}(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})[z_i] + 1$ ▷ Update the count statistics vector
end for
for $i \leftarrow 1, \ldots, n$ **do** ▷ Second pass
    if $i \leq k$ `or` $i \geq n-k+1$ **then**
        $\hat{X}_i = z_i$
    else
        if $Q_i >$ threshold **then** ▷ Quality score
            $\hat{X}_i = z_i$
        else
            $\hat{X}_i(z^n) = \underset{\hat{x}\in\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}}{\arg\min}\ \mathbf{m}^T(z^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})\mathbf{\Pi}^{-1}[\lambda_{\hat{x}} \odot \pi_{z_i}]$ ▷ Apply the denoising rule
        end if
    end if
end for

2. DUDE-Seq does not need to consider reverse complements of the input sequences to collect $\mathbf{m}$'s, since forward and reverse reads are handled separately in our experiments. Reverse complements are typically considered when we need to handle double-stranded sequences without knowing whether each read corresponds to the forward or reverse strand.

2.2.2 Homopolymer Errors

Homopolymer errors, particularly in pyrosequencing, occur while handling the observed flowgram, and a careful understanding of the error injection procedure is necessary to correct these errors. As described in Quince *et al.* [49], in pyrosequencing, the light intensities, *i.e.*, flowgram, that correspond to a fixed order of four DNA bases $\{\mathtt{T}, \mathtt{A}, \mathtt{C}, \mathtt{G}\}$ are sequentially observed. The intensity

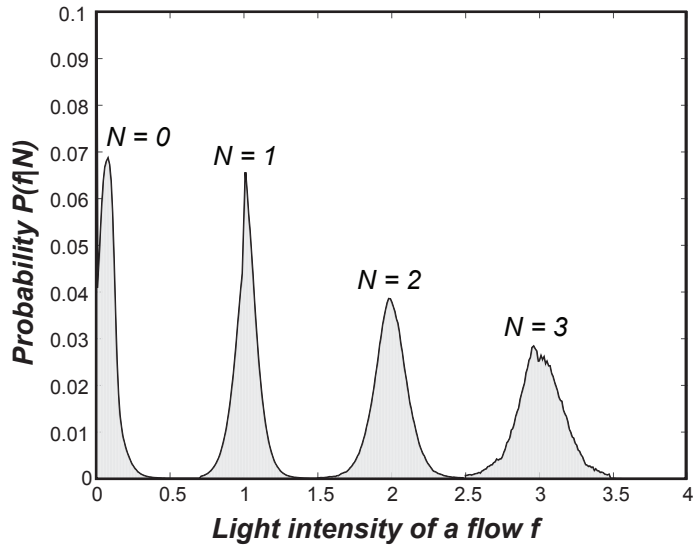


Figure 2.3: Conditional intensity distributions for $N = 0, 1, 2, 3$.

value increases when the number of consecutive nucleotides (*i.e.*, homopolymers) for each DNA base increases, and the standard base-calling procedure rounds the continuous-valued intensities to the closest integers. For example, when the observed light intensities for the two frames of DNA bases are [0.03 1.03 0.09 0.12; 1.89 0.09 0.09 1.01], the corresponding rounded integers are [0.00 1.00 0.00 0.00; 2.00 0.00 0.00 1.00]. Hence, the resulting sequence is `ATTG`. The insertion and deletion errors are inferred because the observed light intensities do not perfectly match the actual homopolymer lengths; thus, the rounding procedure may result in the insertion or deletion of DNA symbols. In fact, the distribution of the intensities f, given the actual homopolymer length N, $\{P(f|N)\}$, can be obtained for each sequencing device, and Figure 2.3 shows typical distributions given various lengths.

Exploiting the fact that the order of DNA bases is always fixed at $\{\mathtt{T}, \mathtt{A}, \mathtt{C}, \mathtt{G}\}$, we can apply the setting of the generalized DUDE in Dembo and Weissman [48] to correct homopolymer errors as follows. Because we know the exact DNA base that corresponds with each intensity value, the goal is the correct estimation of homopolymer lengths from the observed intensity values. Hence, we can

interpret the intensity distributions $\{P(f|N)\}$ as the memoryless noisy channel models with a continuous-output, where the channel input is the homopolymer length N. We set the upper bound of N to 9 according to the convention commonly used for handling flowgram distributions in the targeted amplicon sequencing literature [49, 50, 51]. When the usual rounding function

$$Q_R(f) = \underset{i \in \{0,\ldots,9\}}{\arg\min} |i - f| \tag{2.4}$$

is used as a scalar quantizer, as mentioned above, and the virtual DMC $\mathbf{\Gamma} \in \mathbb{R}^{10\times 10}$ can be obtained by calculating the integral

$$\Gamma(i,j) = \int_{j-0.5}^{j+0.5} P(f|i)df \tag{2.5}$$

for each $0 \leq i \leq 9,\ 1 \leq j \leq 9$ and $\Gamma(i,0) = \int_0^{0.5} P(f|i)df$.

With this virtual DMC model, we apply a scheme inspired by the generalized DUDE to correctly estimate the homopolymer lengths, which results in correcting the insertion and deletion errors. That is, we set $\mathcal{X} = \mathcal{Z} = \{0, 1, \ldots, 9\}$, and again use the Hamming loss $\mathbf{\Lambda} \in \mathbb{R}^{10\times 10}$. With this setting, we apply $Q_R(f)$ to each f_i to obtain the quantized discrete output z_i, and obtain the count statistics vector $\mathbf{m}$ from z^n during the first pass. Then, for the second pass, instead of applying the more involved denoising rule in Dembo and Weissman [48], we employ the same rule as Eq. 2.2 with $\mathbf{\Gamma}$ in place of $\mathbf{\Pi}$ to obtain the denoised sequence of integers $\hat{X}^n$ based on the quantized noisy sequence Z^n. Although it is potentially suboptimal compared to the generalized DUDE, this scheme is used because its implementation is easier and it has a faster running time than that of the generalized DUDE. Once we obtain $\hat{X}^n$, from the knowledge of the DNA base for each i, we can reconstruct the homopolymer error-corrected DNA sequence $\hat{D}$ (the length of which may not necessarily be equal to n). Algorithm 2 summarizes the pseudo-code of DUDE-Seq for homopolymer-error correction.

Algorithm 2 The DUDE-Seq for homopolymer errors

Require: Flowgram data f^n, Flowgram densities $\{P(f|N)\}_{N=0}^{9}$, Hamming loss $\mathbf{\Lambda} \in \mathbb{R}^{10\times 10}$, Context size k

Ensure: The denoised sequence $\hat{D}$

Let $Q_R(f)$ be the rounding quantizer in Eq. 2.4 of the main text
Let $\texttt{Base}(i) \in \{\texttt{T}, \texttt{A}, \texttt{C}, \texttt{G}\}$ be the DNA base corresponding to f_i
Define $\mathbf{m}(f^n, l^k, r^k) \in \mathbb{R}^{10}$ for all $(l^k, r^k) \in \{0, 1, \ldots, 9\}^{2k}$.
Initialize $\mathbf{m}(f^n, l^k, r^k)[a] = 0$ for all $(l^k, r^k) \in \{0, 1, \ldots, 9\}^{2k}$ and for all $a \in \{0, 1, \ldots, 9\}$
Let $\hat{D} = \phi$, $I = 0$
for $i \leftarrow 0, \ldots, 9$ **do**
    for $j \leftarrow 0, \ldots, 9$ **do**
        Compute $\Gamma(i, j)$ following Eq. 2.5 of the main text ▷ Computing the virtual DMC $\mathbf{\Gamma}$
    end for
end for
for $i \leftarrow 1, \ldots, n$ **do** Obtain $z_i = Q_R(f_i)$ ▷ Note $z_i \in \{0, \ldots, 9\}$
end for
for $i \leftarrow k+1, \ldots, n-k$ **do** ▷ First pass
    $\mathbf{m}(f^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})[z_i] = \mathbf{m}(f^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})[z_i] + 1$
end for
for $i \leftarrow 1, \ldots, n$ **do** ▷ Second pass
    if $i \leq k$ `or` $i \geq n-k+1$ **then** $\hat{X}_i(f^n) = z_i$
    else
        $\hat{X}_i(f^n) = \arg\min_{\hat{x}\in\mathcal{X}} \mathbf{m}^T(f^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})\mathbf{\Gamma}^{-1}[\lambda_{\hat{x}} \odot \gamma_{z_i}]$ ▷ Note $\hat{X}_i(z^n) \in \{0, \ldots, 9\}$
    end if
    if $\hat{X}_i(f^n) \geq 1$ **then**
        for $j \leftarrow 1, \ldots, \hat{X}_i(f^n)$ **do** $\hat{D}_{I+j} = \texttt{Base}(i)$ ▷ Reconstructing the DNA sequence
        end for
    end if
    $I \leftarrow I + \hat{X}_i(f^n)$
end for

2.3 Experimental Results

2.3.1 Experiment Setup

We used both real and simulated NGS datasets and compared the performance of DUDE-Seq with that of several state-of-the-art error correction methods. The list of alternative tools used for comparison and the rationale behind our choices are described in the next subsection. When the flowgram intensities of base-calling were available, we corrected both homopolymer and substitution errors; otherwise, we only corrected substitution errors. The specifications of the machine we used for the analysis are as follows: Ubuntu 12.04.3 LTS, 2× Intel Xeon X5650 CPUs, 64 GB main memory, and 2 TB HDD.

DUDE-Seq has a single hyperparameter k, the context size, that needs to be determined. Similar to the popular k-mer-based schemes, there is no analytical method for selecting the best k for finite data size n, except for the asymptotic order result of $k|\mathcal{X}|^{2k} = o(n/\log n)$ in Weissman *et al.* [22], but a heuristic rule of thumb is to try values between 2 and 8. Furthermore, as shown in Eq. 2.2, the two adjustable matrices, $\mathbf{\Lambda}$ and $\mathbf{\Pi}$, are required for DUDE-Seq. The loss $\mathbf{\Lambda}$ used for both types of errors is the Hamming loss. According to Marinier *et al.* [52], adjusting the sequence length by one can correct most homopolymer errors, which justifies our use of Hamming loss in DUDE-Seq. In our experiments, the use of other types of loss functions did not result in any noticeable performance differences. The DMC matrix $\mathbf{\Pi}$ for substitution errors is empirically determined by aligning each sampled read to its reference sequence, as in Quince *et al.* [49]. Figure 2.4 shows the non-negligible variation in the empirically obtained $\mathbf{\Pi}$'s across the sequencing platforms, where each row corresponds to the true signal x and each column corresponds to the observed noisy signal z. In this setting, each cell represents the conditional probability $P(z|x)$. In our experiments, dataset P1–P8 used $\mathbf{\Pi}$ for GS FLX, Q19–Q31 used $\mathbf{\Pi}$ for Illumina, and S5, A5 used $\mathbf{\Pi}$ for Simulation

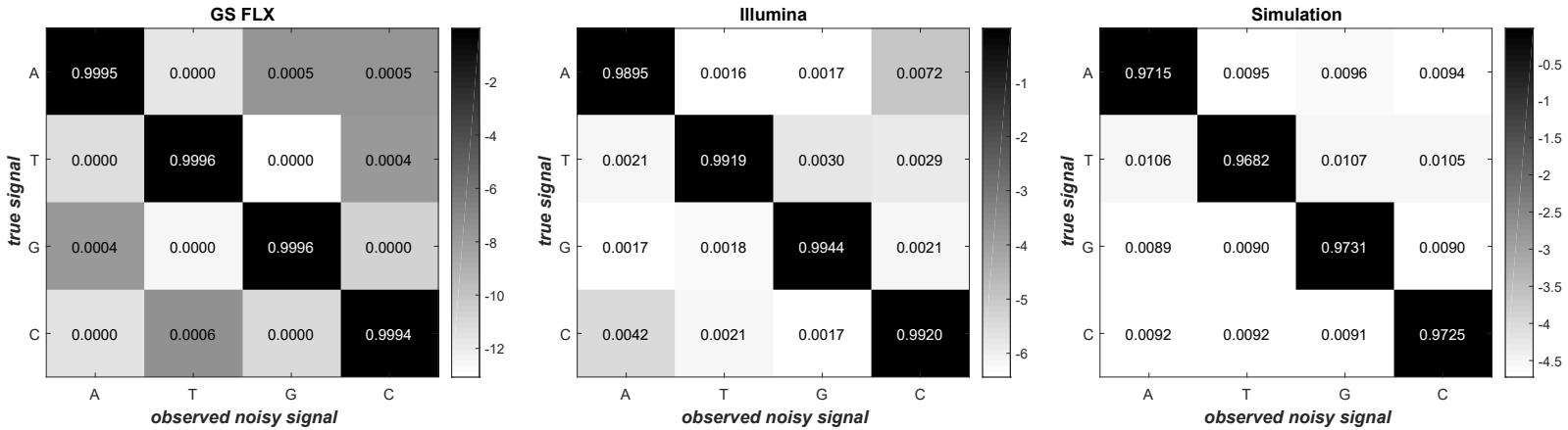


Figure 2.4: Adjustable DMC matrix $\mathbf{\Pi}$ of DUDE-Seq.

data. The details of each dataset are explained in the following sections.

In order to evaluate the results, we used Burrows-Wheeler Aligner (BWA) [53] and SAMtools [54]. We aligned all reads to their reference genome using BWA with the following parameters: [minimum seed length: 19, matching score: 1, mismatch penalty: 4, gap open penalty: 6, gap extension penalty: 1]. After the mapped regions were determined using BWA in SAM format, we chose uniquely mapped pairs using SAMtools. The Compact Idiosyncratic Gapped Alignment Report (CIGAR) string and MD tag (string for mismatching positions) for each of the resultant pairs in the SAM file were reconstructed to their pairwise alignments using sam2pairwise [55].

2.3.2 Evaluation Metric

As a performance measure, we define the per-base error rate of a tool after denoising as

$$e_{\text{tool}} = \frac{\#\text{ mismatched bases}}{\#\text{ aligned bases}}, \tag{2.6}$$

in which '# aligned bases' represents the number of mapped bases (*i.e.*, matches and mismatches) after mapping each read to its reference sequence, and '# mismatched bases' represents the number of the erroneous bases (*i.e.*, insertions, deletions, and substitutions) among the aligned bases.

We also employ an alternative definition that adjusts the error rate by incorporating the degree of alignment. To this end, we define the *relative gain* of the number of aligned bases after denoising by a tool over raw data as

$$g(a_{\text{tool}}) = \frac{\#\text{ aligned bases after denoising} - \#\text{ aligned bases in raw}}{\#\text{ aligned bases in raw}}. \tag{2.7}$$

Based on this, the adjusted error rate $\hat{e}_{\text{tool}}$ of a denoising tool is defined as

follows:

$$\hat{e}_{\text{tool}} = (1 + g(a_{\text{tool}})) \times e_{\text{tool}} - g(a_{\text{tool}}) \times e_{\text{raw}}, \tag{2.8}$$

where e_{tool} and e_{raw} represent the (unadjusted) error rates of the denoised data and the raw data, respectively. In other words, Eq. 2.8 is a weighted average of e_{tool} and e_{raw}, in which the weights are determined by the relative number of aligned bases of a tool compared to the raw sequence. We believe $\hat{e}_{\text{tool}}$ is a fairer measure as it penalizes the error rate of a denoiser when there is a small number of aligned bases. The relative gain of the adjusted error rate over raw data is then defined as

$$g(\hat{e}_{\text{tool}}) = \frac{e_{\text{raw}} - \hat{e}_{\text{tool}}}{e_{\text{raw}}}, \tag{2.9}$$

which we use to evaluate the denoiser performance.

While evaluating a clustering result, we employ a measure of concordance (MoC) [56] which is a popular similarity measure for pairs of clusterings. For two pairs of clusterings P and Q with I and J clusters, respectively, the MoC is defined as

$$\text{MoC}(P, Q) = \frac{1}{\sqrt{IJ} - 1} \left(\sum_{i=1}^{I} \sum_{j=1}^{J} \frac{f_{ij}^2}{p_i q_j} - 1 \right) \tag{2.10}$$

where f_{ij} is the number of the common objects between cluster P_i and Q_j when p_i and q_j are the numbers of the objects in cluster P_i and Q_j, respectively. A MoC of one or zero represents perfect or no concordance, respectively, between the two clusters.

2.3.3 Software Chosen for Comparison

It is impossible to compare the performance of DUDE-Seq with that of all other schemes. Hence, we selected representative baselines using the following

reasoning.

1. We included tools that can represent different principles outlined in the Introduction, namely, k-mer-based (Trowel, Reptile, BLESS, and fermi), MSA-based (Coral), and statistical error model-based (AmpliconNoise) methods.

2. We considered the recommendations of Laehnemann *et al.* [21] to choose baseline tools that are competitive for different scenarios, *i.e.*, for 454 pyrosequencing data (AmpliconNoise), non-uniform coverage data, such as metagenomics data (Trowel, fermi, Reptile), data dominated by substitution errors, such as Illumina data (Trowel, fermi, Reptile), and data with a high prevalence of indel errors (Coral).

3. For multiple k-mer-based tools, we chose those that use different main approaches/data structures: BLESS (k-mer spectrum-based/hash table and bloom filter), fermi (k-mer spectrum and frequency-based/hash table and suffix array), Trowel (k-mer spectrum-based/hash table), and Reptile (k-mer frequency and Hamming graph-based/replicated sorted k-mer list).

4. The selected tools were developed quite recently; Trowel and BLESS (2014), fermi (2012), Coral and AmpliconNoise (2011), and Reptile (2010).

5. We mainly chose tools that return read-by-read denoising results to make fair error-rate comparisons with DUDE-seq. We excluded tools that return a substantially reduced number of reads after error correction (caused by filtering or forming consensus clusters). Examples of excluded tools are Acacia, ALLPATHS-LG, and SOAPdenovo.

6. We also excluded some recently developed tools that require additional mandatory information (e.g., the size of the genome of the reference

organism) beyond the common setting of DNA sequence denoising in order to make fair error-rate comparisons. Examples of excluded tools are Fiona, Blue, and Lighter. Incorporating those tools that require additional information into the DUDE-Seq framework and comparisons with the excluded tools would be another future directions.

2.3.4 Real Data: 454 Pyrosequencing

Pyrosequenced 16S rRNA genes are commonly used to characterize microbial communities because the method yields relatively longer reads than those of other NGS technologies [57]. Although 454 pyrosequencing is gradually being phased out, we tested DUDE-Seq with 454 pyrosequencing data for the following reasons: (1) the DUDE-Seq methodology for correcting homopolymeric errors in 454 sequencing data is equally applicable to other sequencing technologies that produce homopolymeric errors, such as Ion Torrent; (2) using pyrosequencing data allows us to exploit existing (experimentally obtained) estimates of the channel transition matrix $\mathbf{\Gamma}$ (*e.g.*, [49]), which is required for denoising noisy flowgrams by DUDE-Seq (see Algorithm 2); (3) in the metagenomics literature, widely used standard benchmarks consist of datasets generated by pyrosequencing.

In metagenome analysis [58], grouping reads and assigning them to operational taxonomic units (OTUs) (*i.e.*, binning) are essential processes, given that the majority of microbial species have not been taxonomically classified. By OTU binning, we can computationally identify closely related genetic groups of reads at a desired level of sequence differences. However, owing to erroneous reads, nonexistent OTUs may be obtained, resulting in the common problem of overestimating ground truth OTUs. Such overestimation is a bottleneck in the overall microbiome analysis; hence, removing errors in reads before they are assigned to OTUs is a critical issue [49]. With this motivation, in some of our experiments below, we used the difference between the number

of assigned OTUs and the ground truth number of OTUs as a proxy for denoising performance; the number of OTUs was determined using UCLUST [59] at identity threshold of 0.97 which is for species assignment.

We tested the performance of DUDE-Seq with the eight datasets used in Quince *et al.* [49], which are mixtures of 94 environmental clones library from eutrophic lake (Priest Pot) using primers 787f and 1492r. Dataset P1 had 90 clones that are mixed in two orders of magnitude difference while P2 had 23 clones that were mixed in equal proportions. In P3, P4, and P5 and P6, P7, and P8, there are 87 mock communities mixed in even and uneven proportions, respectively. In all datasets, both homopolymer and substitution errors exist, and the flowgram intensity values as well as the distributions are available [49]. Therefore, DUDE-Seq tries to correct both types of errors using the empirically obtained $\mathbf{\Pi}$ and the flowgram intensity distributions $\{P(f|N)\}$.

We first show the effect of k on the performance of DUDE-Seq in Figure 2.5. The vertical axis shows the ratio between the number of OTUs assigned after denoising with DUDE-Seq and the ground truth number of OTUs for the P1, P2, and P8 dataset. The horizontal axis shows the k values used for correcting the substitution errors (*i.e.*, for Algorithm 1), and color-coded curves were generated for different k values used for homopolymer-error correction (*i.e.*, for Algorithm 2). As shown in the figure, correcting homopolymer errors (*i.e.*, with $k = 2$ for Algorithm 2) always enhanced the results in terms of the number of OTUs in comparison to correcting substitution errors alone (*i.e.*, Algorithm 1 alone). We observe that $k = 5$ for Algorithm 1 and $k = 2$ for Algorithm 2 produce the best results in terms of the number of OTUs. Larger k value work better for substitution errors owing to the smaller alphabet size of the data, *i.e.*, 4, compared to that of homopolymer errors, *i.e.*, 10. Motivated by this result, we fixed the context sizes of substitution error correction and homopolymer error correction to $k = 5$ and $k = 2$, respectively, for all subsequent experiments.

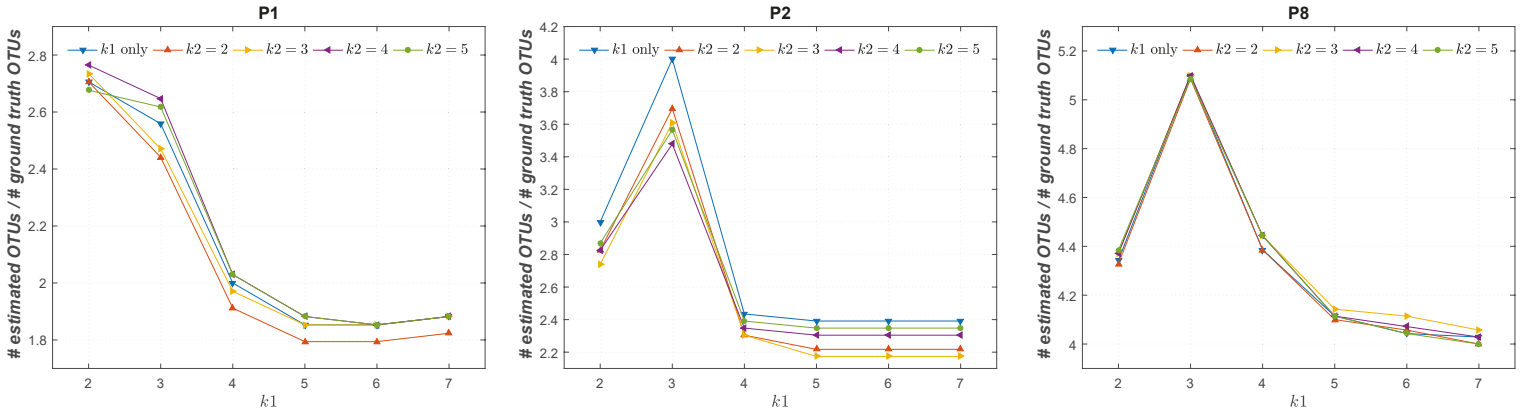


Figure 2.5: Hyperparameter k of DUDE-Seq.

In Figure 2.6(a), we report a more direct analysis of error correction performance. We compared the performance of DUDE-Seq with that of Coral [16], which is an MSA-based state-of-the-art scheme. It aligns multiple reads by exploiting the k-mer neighborhood of each base read and produces read-by-read correction results for pyrosequencing datasets, similar to DUDE-Seq. Furthermore, as a baseline, we also presented the error rates for the original, uncorrected sequences (labeled 'Raw'). We did not include the results of AmpliconNoise [49], a state-of-the-art scheme for 454 pyrosequencing data, in the performance comparison because it does not provide read-by-read correction results, making a fair comparison of the per-base error correction performance with DUDE-Seq difficult. We observeed that DUDE-Seq(1+2), which corrects both substitution errors and homopolymer errors, always outperforms Coral, and the relative error reductions of DUDE-Seq(1+2) with respect to 'Raw,' without any denoising, was up to 23.8%. Furthermore, the homopolymer error correction further drives down the error rates obtained by substitution-error correction alone; hence, DUDE-Seq(1+2) always outperforms DUDE-Seq(1).

In Figure 2.6(b), we compare the error correction performance of three schemes, AmpliconNoise, Coral, and DUDE-Seq, in terms of the MoC. AmpliconNoise assumes a certain statistical model on the DNA sequence and runs an expectation-maximization algorithm for denoising. Here, the two clusterings in the comparison are the golden OTU clusterings and the clusterings returned by denoisers. We observe that for all eight datasets, the number of OTUs generated by DUDE-Seq is consistently closer to the ground truth, providing higher MoC values than those of the other two schemes.

Furthermore, Figure 2.6(c) compares the running time of the three schemes for the eight datasets. We can clearly see that DUDE-Seq is substantially faster than the other two. Particularly, we stress that the running time of DUDE-Seq, even when implemented and executed with a single CPU, is two orders of magnitude faster than that of parallelized AmpliconNoise, run on four

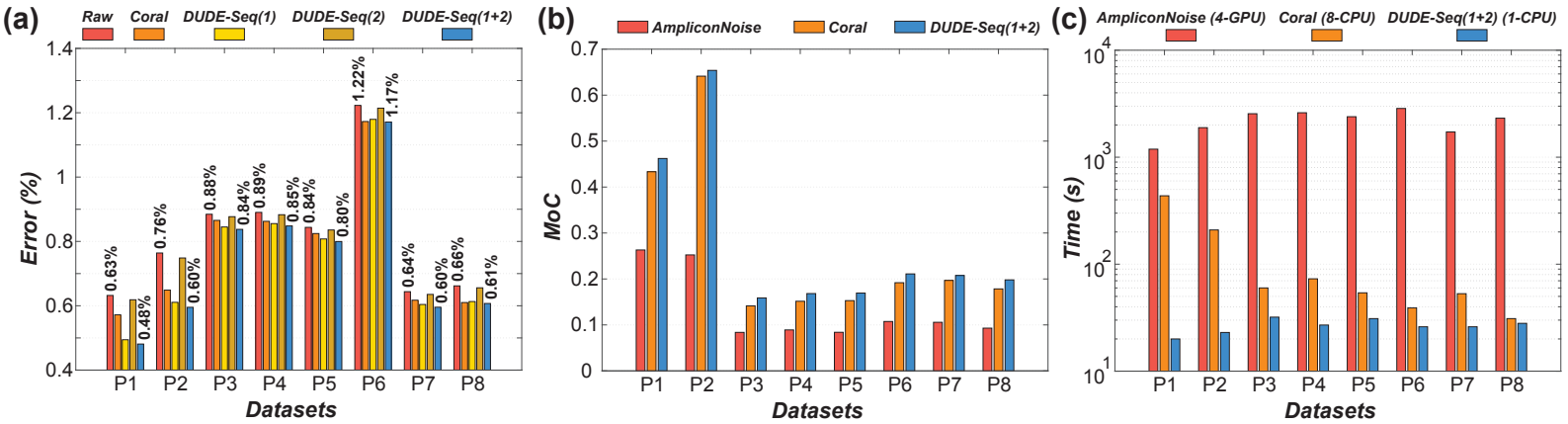


Figure 2.6: Comparison of reads correction performance on eight real 454 pyrosequencing datasets (labeled P1–P8). (a) Per-base error rates [1 and 2 represents substitution error-correction (Algorithm 1) and homopolymer error-correction (Algorithm 2), respectively.] (b) Measure of concordance (MoC), a similarity measure for pairs of clusterings (c) Running time (the type and quantity of processors used for each case are shown in legend)

powerful GPUs. We believe that this substantial boost over state-of-the-art schemes with respect to running time is a compelling reason for the adoption of DUDE-Seq in microbial community analysis.

2.3.5 Real Data: Illumina Sequencing

Illumina platforms, such as GAIIx, MiSeq, and HiSeq, are currently ubiquitous platforms in genome analysis. These platforms intrinsically generate paired-end reads (forward and reverse reads), due to the relatively short reads compared to those obtained by automated Sanger sequencing [60]. Merging the forward and reverse reads from paired-end sequencing yields elongated reads (*e.g.*, 2×300 bp for MiSeq) that improve the performance of downstream pipelines [61].

Illumina platforms primarily inject substitution errors. A realistic error model is not the DMC, though, as the error rates of the Illumina tend to increase from the beginning to the end of reads. Thus, the assumptions under which the DUDE was originally developed do not exactly apply to the error model of Illumina. In our experiments with DUDE-Seq, however, we still used the empirically obtained DMC model $\mathbf{\Pi}$ in Figure 2.4, which was computed by *averaging* all error rates throughout different Illumina platforms.

In our experiments, we used 13 real Illumina datasets (named Q19–Q31) reported previously [46], including sequencing results from four organisms (*Anaerocellum thermophilum Z-1320 DSM 6725*, *Bacteroides thetaiotaomicron VPI-5482*, *Bacteroides vulgatus ATCC 8482*, and *Caldicellulosiruptor saccharolyticus DSM 8903*) targeting two hypervariable regions, V3 and V4, using different configurations (see the caption for Table 2.1 and Figure 2.7 for details). To examine how the number of reads in a dataset affects denoising performance, we derived 10 subsets from the original datasets by randomly subsampling 10,000 to 100,000 reads in increments of 10,000 reads. In addition to Coral, we compared the performance of DUDE-Seq with that of BLESS [62], fermi [63], and Trowel [40], which are representative k-mer-based

Table 2.1: Details of the Illumina datasets used for our experiments shown in Figure 2.7

dataset ID	region	sequencer	Taq	organism	forward & reverse primer
Q19	V4	MiSeq2	Q5	AT	515 & 805RA
Q20	V4	MiSeq2	Q5	BT	515 & 805RA
Q21	V4	MiSeq2	Q5	BV	515 & 805RA
Q22	V4	MiSeq2	Q5	CS	515 & 805RA
Q23	V4	MiSeq2	HF	AT	515 & 805RA
Q24	V4	MiSeq2	HF	BT	515 & 805RA
Q25	V4	MiSeq2	HF	BV	515 & 805RA
Q26	V4	MiSeq2	HF	CS	515 & 805RA
Q27	V3/V4	MiSeq1	Q5	AT	314f & 806rcb
Q28	V3/V4	MiSeq1	Q5	BT	314f & 806rcb
Q29	V3/V4	MiSeq1	Q5	BV	314f & 806rcb
Q30	V3/V4	MiSeq1	Q5	CS	314f & 806rcb
Q31	V3/V4	MiSeq1	HF	AT	314f & 806rcb

Taqs: HiFI Kapa (HF), Q5 neb (Q5); Organisms: Anaerocellum thermophilum Z-1320 DSM 6725 (AT), Bacteroides thetaiotaomicron VPI-5482 (BT), Bacteroides vulgatus ATCC 8482 (BV), Caldicellulosiruptor saccharolyticus DSM 8903 (CS), Herpetosiphon aurantiacus ATCC 23779 (HA), Rhodopirellula baltica SH 1 (RBS), Leptothrix cholodnii SP-6 (LC)

state-of-the-art tools. BLESS corrects "weak" k-mers that exist between consecutive "solid" k-mers, assuming that a weak k-mer has only one error. Fermi corrects sequencing errors in underrepresented k-mers using a heuristic cost function based on quality scores and does not rely on a k-mer occurrence threshold. Trowel does not use a coverage threshold for its k-mer spectrum and iteratively boosts the quality values of bases after making corrections with k-mers that have high quality values.

Figure 2.7 shows the per-base error rates, defined in Eq. 2.6, for the tools under comparison using the first eight datasets (Q19–Q26) and their subsets created as described above (thus, a total of 80 datasets per tool). BLESS did not run successfully on these datasets, and hence its results are not shown. First, we can confirm that DUDE-Seq is effective in reducing substitution errors for data obtained using the Illumina platform in all tested cases of targeted amplicon sequencing, with relative error rate reductions of 6.40–49.92%, compared to the 'Raw' sequences. Furthermore, among the tools included in

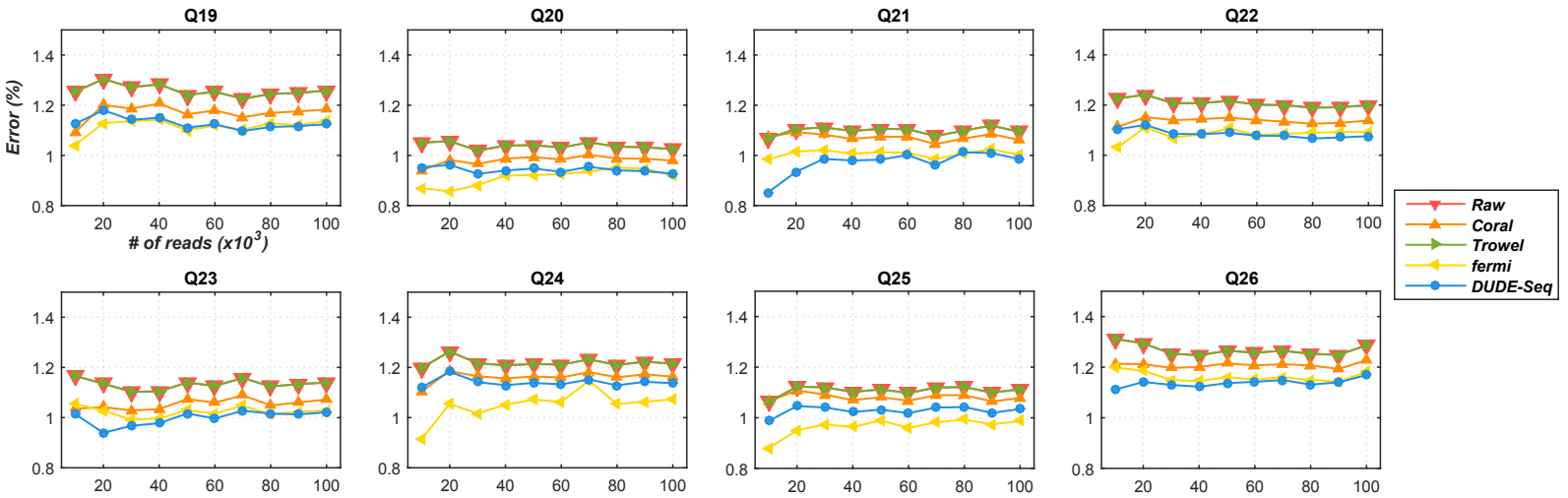


Figure 2.7: Comparison of reads correction performance on real Illumina datasets (labeled Q19–Q26; see Table 2.1 for more details).

the comparison, DUDE-Seq produced the best results for the largest number of datasets. For Q24 and Q25, fermi was most effective, but was outperformed by DUDE-Seq in many other cases. Coral was able to denoise to some extent but was inferior to DUDE-Seq and fermi. Trowel gave unsatisfactory results in this experiment.

Before presenting our next results, we note that while the error rate defined in Eq. 2.6 is widely used for DNA sequence denoising research as a performance measure, it occasionally misleading and cannot be used to fairly evaluate the performance of denoisers. This is because only errors at aligned bases are counted in the error rate calculation; hence, a poor denoiser may significantly reduce the number of aligned bases, potentially further corrupting the noisy sequence, but it can have a low error rate calculated as in Eq. 2.6. In our experiments with the datasets Q27-Q31, we detected a large variance in the number of aligned bases across different denoising tools; thus, it was difficult to make a fair comparison among the performance of different tools with Eq. 2.6. We note that in the experiments presented in Figure 2.6(a) and Figure 2.7, such a large variance was not detected. To alleviate this issue, we employ the alternative definition of the per-base error rate of a tool in Eq. 2.8.

Figure 2.8 shows the results obtained for 100,000-read subsets of each of the Q19–Q31 datasets, *i.e.*, all datasets, for DUDE-Seq and the four alternative denoisers. Because the datasets Q27–Q31 had two subsets of 100,000 reads, we used a total of 18 datasets to draw Figure 2.8, one each from Q19–Q26 and two each from Q27–Q31. As mentioned previously, BLESS could not run successfully on Q19–Q26; hence, there are only 10 points for BLESS in the plots. Figure 2.8(a), (b) and (c) presents the distribution of $g(\hat{e}_{\text{tool}})$, $g(a_{\text{tool}})$, and running times for each tool, respectively. For each distribution, the average value is marked with a solid circle. As shown in Figure 2.8(b), we clearly see that Coral and Trowel show a large variance in the number of aligned bases. For example, Coral only aligns 30% of bases compared to the

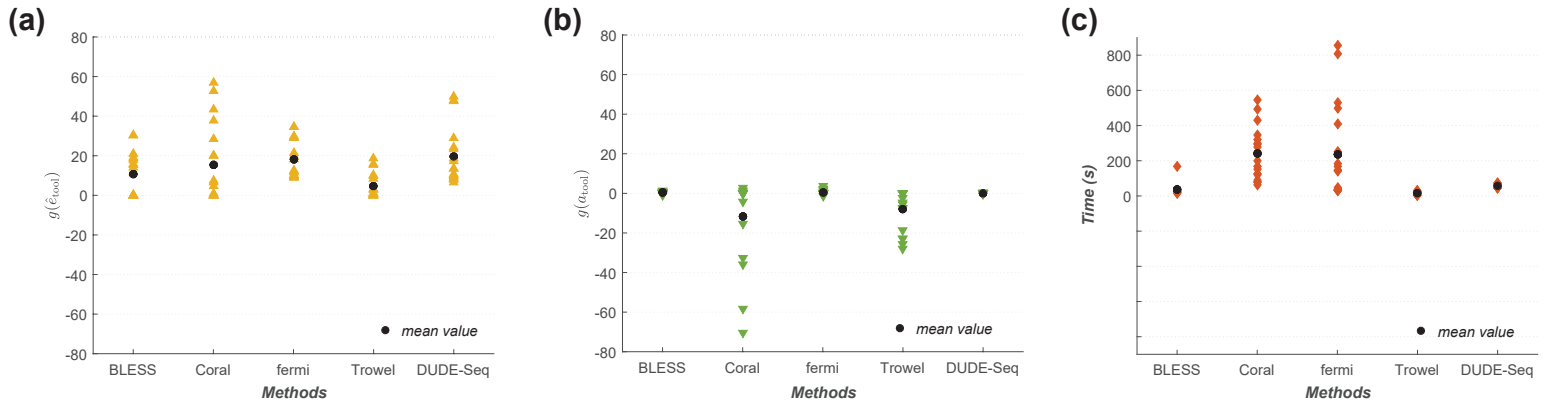


Figure 2.8: Performance comparison. (a) Relative gain of adjusted error rates over 'Raw' data (Eq. 2.9). (b) Relative gain of aligned bases (Eq. 2.7). (c) Running time on real Illumina datasets (labeled Q19–Q31; see the caption for Fig 2.7).

raw sequence after denoising for some datasets. With the effect of this variance in aligned bases adjusted, Figure 2.8(a) shows that DUDE-Seq produces the highest average $g(\hat{e}_{\text{tool}})$, *i.e.*, 19.79%, among all the compared tools. Furthermore, the variability of $g(a_{\text{tool}})$ was the smallest for DUDE-Seq, as shown in Figure 2.8(b), suggesting its robustness. Finally, in Figure 2.8(c), we observe that the running times were significantly shorter for DUDE-Seq and Trowel than for Coral and fermi. Overall, we can conclude that DUDE-Seq is the most robust tool, with a fast running time and the highest average accuracy after denoising.

We performed two statistical tests followed by Shapiro-Wilk test [64] according to its result: one sample t-test or Wilcoxon signed rank test [65]. If the distribution of the sampled results follows a normal distribution according to Shapiro-Wilk test (*i.e.*, $p > 0.05$), we performed one sample t-test otherwise we did Wilcoxon signed rank test. The resulting p-values are listed in Table 2.2. The bold blue ($p < 0.05$) and blue ($p > 0.05$) marked cases represent that DUDE-Seq has better performance than the alternative while the red marked ones do the opposite. While considering $g(\hat{e}_{\text{tool}})$ which embodies $g(a_{\text{tool}})$, we can conclude that most of our experimental results showed statistical significance of differences than the alternatives.

In summary, we observe from Figure 2.7 and 2.8 that DUDE-Seq robustly outperforms the competing schemes for most of the datasets tested. We specifically emphasize that DUDE-Seq shows a strong performance, even though the DMC assumption does not hold for the sequencer. We believe that the better performance of DUDE-Seq relative to other state-of-the-art algorithms (based on MSA or k-mer spectrums) on real Illumina datasets strongly demonstrates the competitiveness of DUDE-Seq as a general DNA sequence denoiser for targeted amplicon sequencing.

Table 2.2: Statistical significance of DUDE-Seq in terms of p-value on Fig 2.8.

		Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26
BLESS	$g(\hat{e}_{\text{tool}})$	-	-	-	-	-	-	-	-
Coral		**2.889e-11**	**7.874e-10**	**1.074e-10**	**1.925e-11**	**5.702e-11**	**2.347e-09**	**1.791e-10**	**1.701e-10**
fermi		0.4325	0.3270	**0.0258**	**0.0047**	**0.0251**	0.0313	8.091e-06	**0.0085**
Trowel		**2.892e-11**	**7.869e-10**	**1.074e-10**	**1.922e-11**	**5.694e-11**	**2.347e-09**	**1.792e-10**	**1.700e-10**

		Q27	Q28	Q29	Q30	Q31
BLESS	$g(\hat{e}_{\text{tool}})$	**0.0488**	0.0241	**0.0121**	0.1309	**4.2767e-08**
Coral		**0.0020**	**0.0195**	**0.0020**	**3.9368e-13**	**0.0020**
fermi		0.2324	0.1444	**0.0273**	**6.0194e-04**	**0.0039**
Trowel		**0.0020**	**4.1690e-10**	**0.0020**	**8.4700e-13**	**0.0020**

The bold blue ($p < 0.05$) and blue ($p > 0.05$) marked cases represent that DUDE-Seq has better performance than the alternative while the red marked ones do the opposite.

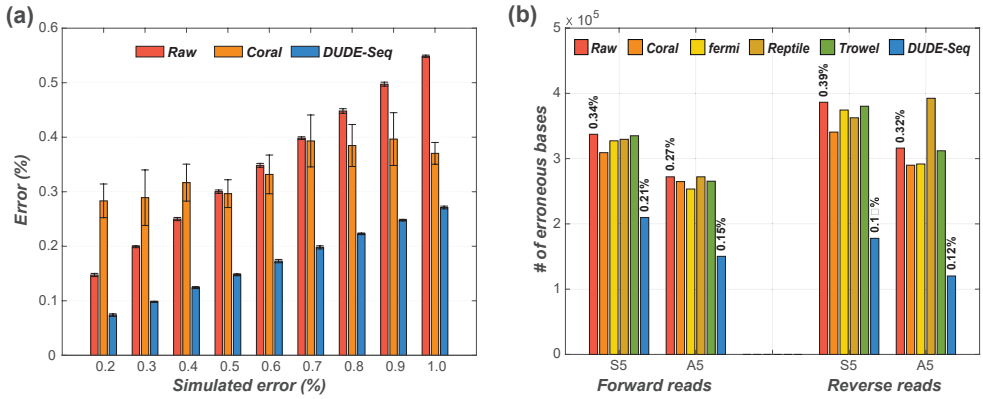


Figure 2.9: Reads correction performance on simulated dataset. (a) Varying error rates using the Grinder simulator. (b) Varying reads composition using the GemSIM simulator (values on top of each bar represent the error rates).

2.3.6 Experiments on Simulated Data

We performed more detailed experiments using Illumina simulators in order to further highlight the strong denoising performance of DUDE-Seq, including the effects on downstream analyses.

Figure 2.9(a) shows the results obtained using the Grinder simulator [66] and a comparison with Coral. Trowel and Reptile require quality scores as input, which are provided by the GemSIM simulator, but not by the Grinder simulator; hence, we could not include Trowel and Reptile in Figure 2.9(a). We generated nine synthetic datasets of forward reads that had error rates at the end of the sequence varying from 0.2% to 1.0%, as denoted on the horizontal axis. For all cases, the error rate at the beginning of the sequence was 0.1%. We again used the *average* DMC model for the entire sequence for DUDE-Seq. Note that the error rates for the 'Raw' data, *i.e.*, the red bars, match the average of the error rates at the beginning and the end of the sequence. From the figure, consistent with the real datasets analyzed in Section 2.3.5, we clearly see that DUDE-Seq significantly outperforms Coral for all tested error rates.

To evaluate the performance of DUDE-Seq for paired-end reads, we gen-

Table 2.2: Details of the public data used for our experiments on simulated data shown in Table 2.3

dataset ID	# total reads	# refs	fragment length	read length	overlap length	simulator (error model)
S5	1,000,000	1	160	100	40	GemSIM (v5[♯])
A5	1,000,000	23	160–190	100	10–40	GemSIM (v5[♯])

[♯] Error model v5 (forward rate 0.28%, reverse 0.34%)

erated datasets, shown in Table 2.2, with the GemSIM sequencing data simulator [67]. As shown in the table, we used 23 public reference sequences [49] to generate the dataset A5 and a single reference sequence for S5. We used the error model v5 that has error rates of 0.28% for forward reads and 0.34% for reverse reads. In Figure 2.9(b), in addition to DUDE-Seq, Coral, fermi, and Trowel, we included the results obtained using Reptile [20], another k-mer spectrum-based method that outputs read-by-read denoising results. We again observe from the figure that DUDE-Seq outperforms the alternatives by significant margins.

In Table 2.3, we show that the error-corrected reads produced by DUDE-Seq can also improve the performance of downstream pipelines, such as paired-end merging. We applied four different paired-end merging schemes, CASPER [68], COPE [69], FLASH [61], and PANDAseq [70], for the two datasets A5 and S5 in Table 2.2. The metrics are defined as usual. A true positive (TP) is defined as a merge with correct mismatching resolution in the overlap region, and a false positive (FP) is defined as a merge with incorrect mismatching resolution in the overlap region. Furthermore, a false negative (FN) is a merge that escapes the detection, and a true negative (TN) is defined as a correct prediction for reads that do not truly overlap. The accuracy and F1 score are computed based on the above metrics [71]. For each dataset, we compared the results for four cases: performing paired-end merging without any denoising, after correcting errors with Coral, after correcting errors with fermi, and after correcting errors with DUDE-Seq. Reptile and Trowel were not included

Table 2.3: Paired-end reads merging performance statistics

tool	dataset	# merges	TP	FP	FN	accuracy	F_1
CASPER	S5	1,000,000	997,303	2,697	**0**	0.997	0.999
COPE		974,219	961,366	12,853	25,781	0.961	0.980
FLASH		999,921	977,431	22,490	79	0.977	0.989
PANDAseq		999,947	976,807	23,140	53	0.977	0.988
CASPER	S5 w/ Coral	1,000,000	997,510	2,490	**0**	0.998	0.999
COPE		975,803	963,717	12,086	24,197	0.964	0.982
FLASH		999,942	978,835	21,107	58	0.979	0.989
PANDAseq		999,949	978,270	21,679	51	0.978	0.989
CASPER	S5 w/ fermi	1,000,000	997,356	2,644	**0**	0.997	0.999
COPE		994,025	969,451	24,574	**5,975**	0.969	0.984
FLASH		999,933	972,025	27,908	67	0.972	0.986
PANDAseq		999,952	971,567	28,385	48	0.972	0.986
CASPER	S5 w/ DUDE-Seq	1,000,000	**999,320**	**680**	**0**	**0.999**	**1.000**
COPE		987,238	**983,639**	**3,599**	12,762	**0.984**	**0.992**
FLASH		999,958	**992,915**	**7,043**	**42**	**0.993**	**0.996**
PANDAseq		999,949	**991,146**	**8,803**	**51**	**0.991**	**0.996**
CASPER	A5	999,973	997,202	2,771	27	0.997	**0.999**
COPE		924,634	915,981	8,653	75,366	0.916	0.956
FLASH		999,578	977,355	22,223	422	0.977	0.989
PANDAseq		999,122	978,720	20,402	878	0.979	0.989
CASPER	A5 w/ Coral	999,974	995,899	4,075	**26**	0.996	0.998
COPE		927,757	918,733	9,024	72,243	0.919	0.958
FLASH		999,742	978,814	20,928	**258**	0.979	0.989
PANDAseq		999,351	979,899	19,452	649	0.980	0.990
CASPER	A5 w/ fermi	999,969	997,288	2,681	31	0.997	**0.999**
COPE		939,986	923,252	16,734	60,014	0.923	0.960
FLASH		999,732	974,903	24,829	268	0.975	0.987
PANDAseq		999,328	975,320	24,008	672	0.975	0.988
CASPER	A5 w/ DUDE-Seq	999,971	**998,078**	**1,893**	29	**0.998**	**0.999**
COPE		943,531	**939,555**	**3,976**	**56,469**	**0.940**	**0.969**
FLASH		999,638	**989,860**	**9,778**	362	**0.990**	**0.995**
PANDAseq		999,354	**989,250**	**10,104**	**646**	**0.989**	**0.995**

[parameters: $k = 5$ for DUDE-Seq; $(k, mr, mm, g) = (21, 1, 1, 1000)$ for Coral; $(k, O, C, s) = (21, 3, 0.3, 5)$ for fermi]

in this experiment because they were generally outperformed by Coral and fermi, as shown in Figure 2.9(b). The accuracy and F1 score results show that correcting errors with DUDE-Seq consistently yields better paired-end merging performance, not only compared to the case with no denoising, but also compared to the cases with Coral and fermi error correction. This result highlights the potential application of DUDE-Seq for boosting the performance of downstream DNA sequence analyses.

2.4 Discussion

Our experimental results show that DUDE-Seq can robustly outperform k-mer-based, MSA-based, and statistical error model-based schemes for both real-world datasets, such as 454 pyrosequencing and Illumina data, and simulated datasets, particularly for targeted amplicon sequencing. This performance advantage in denoising further allowed us to obtain improved results in downstream analysis tasks, such as OTU binning and paired-end merging. Furthermore, the time demand of DUDE-Seq-based OTU binning is order(s) of magnitude lower than that of the current state-of-the-art schemes. We also demonstrated the robustness and flexibility of DUDE-Seq by showing that a simple change in $\mathbf{\Pi}$ matrix is enough to apply the exact same DUDE-Seq to data obtained using different sequencing platforms. In particular, we experimentally showed that even when the memoryless channel assumption does not hold, as in Illumina data, DUDE-Seq still solidly outperforms state-of-the-art schemes.

The sliding window nature of DUDE-Seq resembles the popular k-mer-based schemes in the literature. However, while all existing k-mer-based schemes rely on heuristic threshold selection for determining errors in the reads, regardless of the error model of the sequencing platform, DUDE-Seq applies an analytic denoising rule that explicitly takes the error model $\mathbf{\Pi}$ into account. Therefore, even for identical noisy reads z^n, DUDE-Seq may result in different denoised sequences, depending on the $\mathbf{\Pi}$'s of different sequencing platforms, whereas the k-mer-based scheme will always result in the exact same denoised sequence. The performance gains reported in this paper compared to state-of-the-art baselines, including those for k-mer-based schemes, substantiate the competitiveness of our method for targeted amplicon sequencing.

Another advantage of DUDE-Seq is its read-by-read error-correction capability. This feature is important for a number of bioinformatics tasks, including *de novo* sequencing, metagenomics, resequencing, targeted resequencing,

and transcriptome sequencing, which typically require the extraction of subtle information from small variants in each read. In addition to the types of tasks presented in this paper (*i.e.*, per-based error correction, OTU binning, and paired-end merging), we plan to apply DUDE-Seq to additional tasks, as mentioned above.

Additional venues for further investigation include the procedure for estimating the noise mechanism represented by $\mathbf{\Pi}$, which is currently empirically determined by aligning each read to the reference sequence and is therefore sensitive to read mapping and alignment. For more robust estimation, we may employ an expectation-maximization-based algorithm, as was recently proposed for estimating substitution emissions for the data obtained using nanopore technology [72]. Considering uncertainties in $\mathbf{\Pi}$ may also be helpful; hence, it may be useful to investigate the relevance of the framework in Gemelos *et al.* [73]. Additionally, it will likely be fruitful to utilize the information in Phred quality scores to make decisions about noisy bases and to fine-tune the objective loss function in our approach. Using a lossy compressed version of the quality scores is one possible direction for boosting the inferential performance of some downstream applications, as shown in Ochoa *et al.* [74]. Furthermore, particularly for the homopolymer error correction, there are several hyperparameters whose choices can be experimented with in the future to potentially achieve substantial performance boosts. Examples include the choice of alphabet size (in lieu of the current value of 10), the choice of the loss function that may be proportional to the difference between the true and estimated value of N (in lieu of the current Hamming loss), and the choice of quantization (in lieu of Eq. 2.4). Moreover, we may apply the full generalized DUDE in Dembo and Weissman [48] for homopolymer error correction to determine if better performance can be achieved at the cost of increased complexity. Applying DUDE-Seq to other types of sequencing technology with homopolymer errors (*e.g.*, Ion Torrent) would also be possible as long as we can acquire flow

(*e.g.*, ionogram) density distributions to estimate $\mathbf{\Gamma}$. Currently, there exists no public data repository that includes such information for Ion Torrent, and thus existing Ion Torrent denoisers often ignore homopolymer errors or rely on simplistic noise modeling and iterative updates that unrealistically limit the maximum length of homopolymer errors that can be handled, let alone computational efficiency [50].

2.5 Summary

We have considered the correction of errors from nucleotide sequences produced by next-generation targeted amplicon sequencing. The NGS platforms can provide a great deal of sequencing data thanks to their high throughput, but the associated error rates often tend to be high. Denoising in high-throughput sequencing has thus become a crucial process for boosting the reliability of downstream analyses. Our methodology, named DUDE-Seq, is derived from a general setting of reconstructing finite-valued source data corrupted by a discrete memoryless channel and effectively corrects substitution and homopolymer indel errors, the two major types of sequencing errors in most high-throughput targeted amplicon sequencing platforms. Our experimental studies with real and simulated datasets suggest that the proposed DUDE-Seq not only outperforms existing alternatives in terms of error-correction capability and time efficiency, but also boosts the reliability of downstream analyses. Further, the flexibility of DUDE-Seq enables its robust application to different sequencing platforms and analysis pipelines by simple updates of the noise model.

Chapter 3

Scalability of a Denoiser

Recent studies have revealed a critical role of the human microbiome in various physiological processes and diseases. For example, the microbes are involved in digestion, production of nutrients, protection against pathogens, and immune homeostasis; their changes in numbers and compositions are associated with the occurrence of numerous diseases, including rheumatoid arthritis, inflammatory bowel disease, colorectal cancer, obesity, cardiovascular disease, and diabetes. As the importance of the human microbiome increases, the urgent need for rapid development of sequencing methods and analytical techniques to study the human microbiome is increasing.

Metagenomics refers to analyzing the functions and sequences of the collective microbial genomes contained in an environmental sample [4]. In our body, the microbes collectively make up to 100 trillion cells, tenfold the number of human cells, and mostly reside in the intestine [75]. Since there exists substantial diversity of the gut microbiome between individuals [76], understanding the diversity by metagenomics may lead to breakthroughs in personalized medicine and related industry [77]. Metagenomic studies typically produce a huge amount of sequence data (*e.g.*, [75] generated 576.7 gigabases in total) due to the microbial diversity. Without data mining, it would be nearly impossible to analyze such large-scale data.

NGS has revolutionized metagenomic research by facilitating high-throughput cataloging of microbial samples that are refractory to analysis by classical culture-based techniques [78]. Compared to automated Sanger method or chain-termination method, NGS methods have much shorter read-length, making genome assembly and annotation more challenging to solve computationally [79].

In many applications, a homologous DNA region such as well-conserved 16S rRNA genes from diverse microbiomes is first amplified by polymerase chain reaction (PCR) before sequenced. Pyrosequencing has allowed much larger read numbers from PCR amplicons than ever before [27]. Even so, pyrosequenced reads are never free from error, and a robust data-mining method is required to remove such error. Otherwise, the noise introduced during PCR and pyrosequencing may lead to overestimating the number of operational taxonomic units (OTUs) by orders of magnitude [80].

3.1 Background

3.1.1 Flowgrams bear Sequence Information

Figure 3.1 shows a procedure to determine the order of nucleotide bases in pyrosequencing. The sequence of a DNA fragment (*i.e.*, sequencing read) is determined by a flowgram F which consists of a series of flows f (*i.e.*, light intensity). A flowgram has four consecutive flows in the order of nucleotides `T`, `C`, `A`, and `G`, respectively. The process of assigning the corresponding nucleotide bases from a flowgram is called base-calling. Let $F = (f_1, f_2, \cdots, f_m)$ represents a flowgram of m flows where $f_i \in \mathbb{R}_+$. A widely used convention for base-calling the flowgram is the number of nucleotides (*i.e.*, lengths) for each base to have $\lfloor f_i + 0.5 \rfloor$ nucleotides. A value above 1.0 corresponds to a homopolymer.

The distance between flowgram F and G of m intensities is defined as

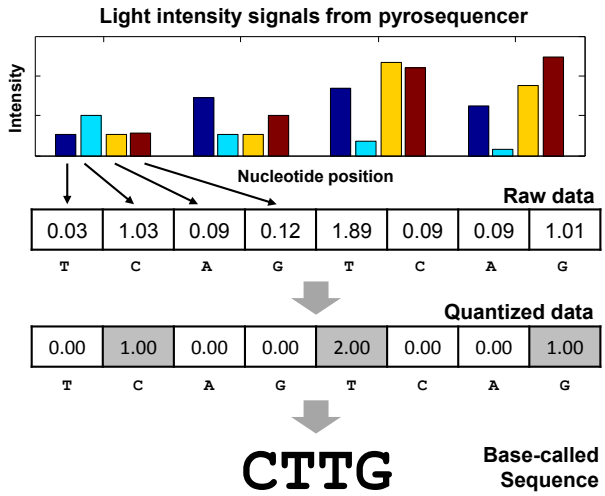


Figure 3.1: Flowgrams and base-calling procedure.

follows [25]:

$$d(F,G) = \frac{1}{m}\sum_{i=1}^{m}\delta(f_i, g_i) \tag{3.1}$$

where $\delta(\cdot,\cdot)$ is an empirical bivariate probability distribution representing the distance between two flows. Based on this distance, the empirical probability that flowgram F and G originate from the same fragment is defined as follows [25]:

$$P(F=G) = \frac{1}{\sigma}\exp\left[-\frac{d(F,G)}{\sigma}\right] \tag{3.2}$$

where σ is a user-specified parameter. This probability decreases exponentially as their distance increases.

3.1.2 Noise Sources in Pyrosequenced Amplicons

As mentioned in the previous sections, a homologous region of a gene is amplified by PCR and then pyrosequenced in microbiome analysis. Both the amplification and pyrosequencing processes cause noises, which distort the analysis result. There are three sources of noise in the pyrosequenced amplicons [27]: base-calling, PCR single base substitutions, and PCR chimeras. The base-calling error refers to the incorrect estimation of homopolymer lengths as

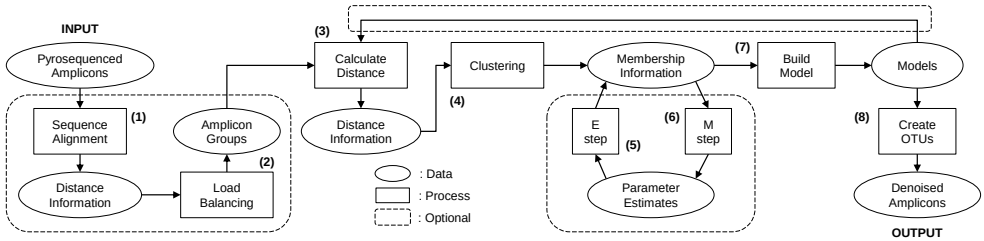


Figure 3.2: Generalized overview of removing noise in pyrosequenced amplicons.

the light intensities of flows do not exactly reflect the homopolymer lengths. The PCR single base substitution refers to a base which is erroneously substituted with another nucleotide (*e.g.*, nucleotide `G` is substituted with nucleotide `C`). This study focuses on these two sources to denoise the sequencing reads because specific knowledge of non-chimeric sequences is usually required to handle PCR chimeras and it is difficult to be generalized.

3.1.3 Existing Denoisers for Pyrosequenced Amplicons

To remove noise from the reads obtained using NGS, various approaches have been proposed. Table 3.1 lists the currently available denoisers, their procedures, and the types of noise they target. Most of these methods handle either the base-calling or PCR single base substitution noises and have common internal procedures, as annotated in Table 3.1 and Figure 3.2. In contrast to other methods, AmpliconNoise handles both types of noise by iterating the procedures shown in Fig. 3.2.

Of note is that, although the sequence of steps in Acacia (target noise "B" or base-calling error) seems identical to that in CRiSPy-CUDA or CUDA-EC (target noise "P" or PCR substitution error), their input types are different. That is, the base-calling error correction can be considered as analog-to-digital conversion, and Acacia accepts light intensity curves as input and returns a discrete sequence as output. By contrast, CRiSPy-CUDA and CUDA-EC take as input discrete sequences and outputs denoised discrete sequences. In other

Table 3.1: Comparison of related noise-removal methods

method	sequence of steps†	target noise‡	ref.
Acacia	$3, 4, 7, 8$	B	[41]
AmpliconNoise	$1, 2, (3, 4, (5, 6)^*, 7)^2, 8$	B, P	[27]
CRiSPy-CUDA	$3, 4, 7, 8$	P	[23]
CUDA-EC	$3, 4, 7, 8$	P	[24]
DecGPU	Spectral alignment	P	[81]
Denoiser	Greedy algorithm	P	[26]
PyroNoise	$1, 2, 3, 4, (5, 6)^*, 7, 8$	B	[25]
NoDe	$3, 4, 7, 8$	P	[82]

† As labeled in Figure 3.2; $(\cdot)^2$: repeat twice; $(\cdot)^*$: repeat until convergence.

‡ B: base-calling error; P: PCR single base substitutions.

words, the steps (3) and (4) in Acacia handle continuous intensity curves, while the steps (3) and (4) in CriSPy-CUDA and CUDA-EC process discrete sequences.

3.1.4 Profiling AmpliconNoise

Among the existing tools for noise reduction, AmpliconNoise [27] is widely used in the metagenomics community as a *de facto* standard denoising tool. Figure 3.3 shows the overall flow of AmpliconNoise, which consists of four major stages. First, to facilitate the later stages, the input dataset is split into nonoverlapping partitions of similar sizes. In the second step, AmpliconNoise removes pyrosequencing noise that appears as insertion/deletion type errors based on an expectation-maximization (EM) approach [83]. Next, substitution errors occurring in the PCR process are removed by another run of the EM procedure. Finally, based on the denoised sequences, OTUs are derived.

Quince *et al.* [27] described a message-passing interface (MPI)-based parallel implementation of AmpliconNoise. According to our runtime profiling listed in Table 3.2, this implementation is not effective in speeding up the steps to compute pairwise distances between nucleotide sequences (*i.e.*, NDist in the first stage and SeqDist in the third stage). These steps are required for the global alignment of each pair of sequences using the Needleman-Wunsch

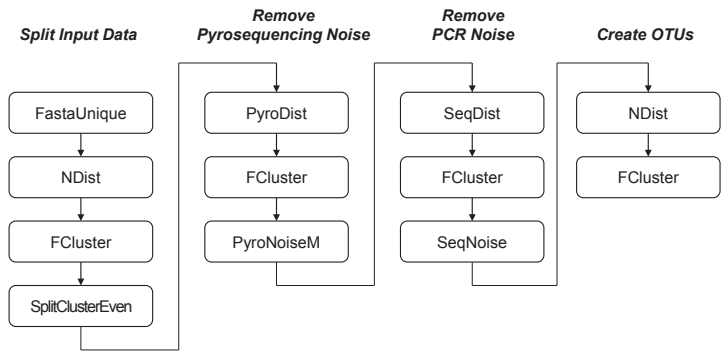


Figure 3.3: Overall flow of AmpliconNoise, consisting of four major stages.

Table 3.2: Runtime profiling of AmpliconNoise

stage	step	time (h:m:s)	remarks
1	FastaUnique	00:00:20	Redundancy removal
	NDist	18:17:32	Distance between sequences
	FCluster	00:30:24	Hierarchical clustering
	SplitClusterEven	00:00:01	Data split
2	PyroDist	00:02:51	Distance between flowgrams
	FCluster	00:06:11	
	PyroNoiseM	00:18:22	Indel-type error removal
3	SeqDist	39:20:30	Distance between sequences
	FCluster	00:11:26	
	SeqNoise	00:11:31	PCR error removal
4	NDist	00:00:04	
	FCluster	00:00:01	OTU estimation

[CPU: Intel Xeon X5650 @ 2.67GHz × 2, RAM: 64-GB DDR3, GPU: NVIDIA GeForce GTX 580 (3-GB GDDR5) × 4]

algorithm [84] and occupy the majority of the total runtime. Based on this profiling result, we decided to parallelize the NDist and SeqDist steps using GPUs, which are suitable for computations with abundant data-level parallelism such as pairwise distance calculation.

3.1.5 CUDA Programming Model

GPUs are specialized for intensive, highly parallel computation by devoting more transistors to data processing rather than to data caching and flow con-

trol. GPUs were originally designed for graphics, and thus they are effective for stream processing that operates in parallel on multiple records in a stream [28]. In an effort to use GPUs for general-purpose computing, we utilized compute unified device architecture (CUDA), which is a parallel computing platform and programming model designed and developed by NVIDIA. CUDA enables heterogeneous programming by exploiting GPUs as a computational device that operates as a coprocessor to the CPU.

3.2 Methods

Figure 3.4 shows the overview of denoising two types of errors in pyrosequenced amplicons, homopolymer and substitution errors. According to Quince *et al.* [25, 27], the overall denoising pipeline consists of two phases of eight procedures using flowgrams and their base-called sequences. In the first phase, the homopolymer errors are denoised using the flowgrams. In the second phase, the substitution errors are denoised by iterating the procedures using the base-called sequences. In each phase, the steps (1)–(2) are to distribute the high-throughput sequencing reads to heterogeneous processors and the steps (1)–(4) are the rate-limiting steps that bottleneck the overall pipeline.

The fundamental idea of this approach is to estimate the clusters in sequencing reads under the assumption that each cluster represents a candidate microbe model. In the first step of denoising homopolymer errors, distances between the flowgrams are calculated. Using the resultant distances, clusters of the neighboring flowgrams are built. The representative flowgrams for each cluster are inferred by maximizing the likelihood of observing all the flowgrams using the expectation-maximization (EM) algorithm [83]. This allows us to estimate the representative flowgrams of each flowgram in the clusters. The base-called sequences of these representative flowgrams are fed to the second phase to denoise the substitution errors in that they are the candidate sequence models. In the first step of denoising substitution errors, distances

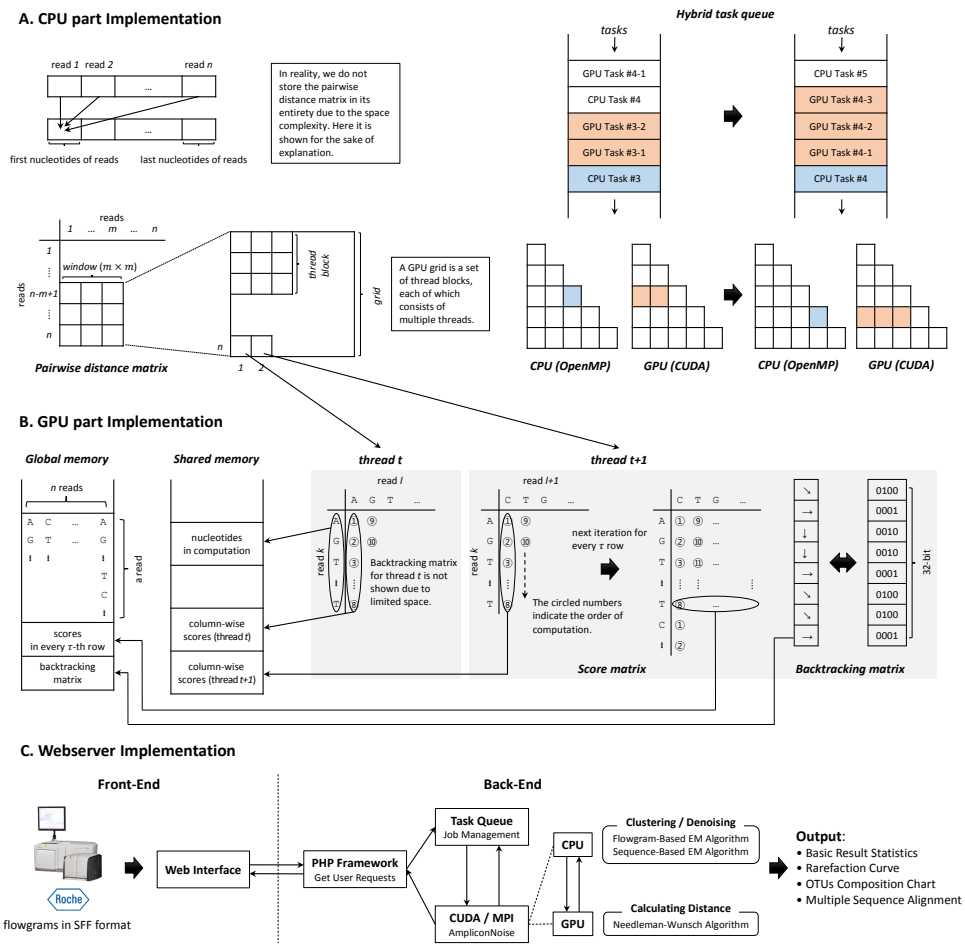


Figure 3.4: Overview of the internal architecture of MUGAN and the GPU-based scheme for parallelizing the denoise process.

between those base-called sequences are calculated. The remainder of the second phase proceeds similar to that of the first one. At the end of the overall pipeline, each denoised sequence is assigned to one of the OTUs to estimate the diversity of microbial community.

Figure 3.4(a) and (b) show the overview of the proposed multi-GPU-based acceleration of the pairwise distance calculation that is optimized for short-read sequencing results. The fundamental concept is to split the global pairwise distance matrix into submatrices and allow each GPU thread to handle each pair. Note that GPUs have a memory hierarchy distinct from that of CPUs and care must be taken in order not to incur excessive latency by suboptimal memory access and transfers [85]. In this regard, our approach employs a number of optimization techniques previously proposed in the literature [86, 87, 88] for dynamic-programming-based sequence alignment on GPUs, such as efficient use of shared memory, coalesced access to global memory, and compact encoding of backtracking information. To accelerate the pairwise distance computation, we adopted our prior work on GPU-based acceleration of distance calculation [89] and extended it to multi-GPU and CPU co-processing. The OpenMP framework was used to create CPU threads, each of which manages the kernel for each local GPU with additional computation on CPUs and MPI for utilizing remote GPUs as well. In other words, our approach utilizes both CPUs and GPUs for computation, and CPUs not only manage GPUs, but also participate in computation.

3.2.1 Multi-GPU-Based Pairwise Distance Computation

The distance between two sequences is defined based on their global alignment using the Needleman-Wunsch algorithm which has quadratic worst-case time complexity. There are two options for parallelizing the global sequence alignment for our denoising algorithm according to the direction of a calculation process. As we process a dynamic programming matrix in diagonal order, the

sequence alignment itself is processed with multiple threads in the kernel. This scheme is generally appropriate for aligning long reads, but in metagenomics, the length of each read is usually short. In this situation, to process a dynamic programming submatrix in the vertical order is recommended as depicted in Figure 3.4. By doing so, multiple threads in the kernel can perform multiple pairwise sequence alignments simultaneously.

Assume that there are n reads whose maximum length is L. Our basic principle of GPU parallelization is to split the $\binom{n}{2}$ independent computations of the pairwise distance matrix into multiple submatrices and assign one GPU grid to each submatrix such that one thread handles one pair of sequences. To achieve the highest occupancy of a GPU, we should maximize the active thread blocks per streaming multiprocessor and better utilize the memory hierarchy while minimizing the host-device communication [85].

Among the various considerations to take into account to fully exploit GPUs, two issues that we considered are as follows. First, a GPU has fewer threads than $\binom{n}{2}$ in case of the high-throughput sequencing which usually has large n. Therefore, we need an efficient divide-and-conquer approach. Second, the global sequence alignment algorithm, which is based on dynamic programming, has a high space complexity to handle both scoring and backtracking matrices simultaneously. A naïve implementation would require $O(2L^2n^2)$ space. Therefore, a scheme for efficient utilization of the hierarchy of GPU memory is necessary.

To implement multi-GPU and CPU co-processing, a CPU thread puts all the tasks (*i.e.*, $m \times m$ submatrices) into a hybrid task queue. According to the user-specified scheduling scheme, for example, we compute the submatrices which have low complexity in the CPUs and the other submatrices in the GPUs. One GPU grid computes all the entries within one submatrix simultaneously, by assigning one thread to compute one distance. The size of each transfer is $O(km^2)$ bytes, where k is the maximum length of the pair-

wise alignment. A GPU thread fills up the score and backtracking matrices by considering τ rows as a unit of computation, where τ is a prime factor of the warp size (*i.e.*, one of 2, 4, 8, 16, and 32 for the warp size of 32). Further details are as follows:

- The k-th nucleotides of the reads are stored in global memory followed by the $(k+1)$-th nucleotides for coalesced memory access.
- For each $L \times L$ matrix, the first $\tau \times L$ submatrix is computed in the first iteration followed by the second submatrix, and so on.
- For each $\tau \times L$ submatrix, the τ nucleotides in computation are stored in shared memory to reduce the access latency of global memory.
- Each $\tau \times L$ submatrix is computed columnwise, namely, the first column is computed followed by the second column, and so on.
- The columnwise $\tau \times 1$ scores in each submatrix are stored in shared memory to be used for computing the next column.
- At the end of each iteration, the τ-th row in each submatrix is stored in global memory to be used for the next iteration.
- Each direction value is encoded into $(32/\tau)$ bits, and then the columnwise $\tau \times 1$ direction values in each submatrix are stored in a 32-bit encoded value in global memory to be transferred to host memory.

The GPU memory utilization explained above was inspired by the prior work on the GPU-based parallelization of dynamic programming [88, 86, 87]. The model construction is done in the host with the pairwise alignment results coming from the device. That is, after the device finishes pairwise alignments of input reads, the host calculates pairwise read distances from the alignment results. Abundant data-level parallelism exists in the problem (*e.g.*, individual pairwise alignment is independent of each other), and thus there are no data

dependencies among each GPU computations. Furthermore, our approach uses multiple GPU cards to improve the scalability and throughput of pairwise distance calculations. To manage multiple GPU kernels simultaneously, we created as many CPU threads as the number of GPU cards in the OpenMP framework and allowed each CPU thread to launch the GPU kernel of each GPU iteratively. The number of kernel runs can be adjusted based on the type of 454 sequencer used and the resulting read length. This CPU-GPU hybrid approach is effective for managing multiple GPUs as well as for maximizing the utilization of each GPU.

3.2.2 Constructing OTU Models

The clusters (*i.e.*, candidate OTUs) in the sequencing reads are estimated for denoising under the assumption that each cluster represents a microbe model. Assume that there are n flowgrams $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ and k candidate OTUs $\mathbf{O} = \{O_1, O_2, \ldots, O_k\}$. The representative flowgrams for the candidate OTUs $\mathbf{O}$ are inferred by clustering the flowgrams $\mathbf{F}$ into k candidate OTUs and then estimating the representative flowgrams $\hat{\mathbf{F}} = \{\hat{F}_1, \hat{F}_2, \ldots, \hat{F}_k\}$.

While estimating the representative flowgrams, each flowgram can be assigned to multiple OTUs (*i.e.*, soft clustering). To assign OTUs for each flowgram, the likelihood of observing the n flowgrams from k candidate OTUs is maximized according to Quince *et al.* [25]:

$$L(\mathbf{F}, \mathbf{Z}|W, \mathbf{O}) = \prod_{i=1}^{n} \prod_{j=1}^{k} [w_j P(F_i = O_j)]^{z_{ij}} \tag{3.3}$$

where $z_{ij} \in [0, 1]$ represents the soft assignment of flowgram i to cluster j and w_j is the weight to cluster j and $\mathbf{Z}$ and W are the set of z_{ij} and w_j, respectively and $P(\cdot)$ is the empirical probability in Eq. 3.2.

To denoise the sequencing reads with this OTU model, we should determine the number of the candidate OTUs k and infer their representative flowgrams

$\hat{\mathbf{F}}$.

Determining the Number of Models

To determine the number of candidate OTUs k, there are various approaches to determine the actual number of clusters in unsupervised learning algorithms [90], which include the following. One approach is to determine the k that maximizes the silhouette coefficient [91] of k-seed-based clustering results. Another approach is to partition the dendrogram of a hierarchical clustering result at a given height. In this study, we adopt the latter approach using the distances determined by Eq. 3.1.

Selecting Model Flowgrams

Given the estimated parameters W and $\mathbf{O}$, we want to maximize the likelihood in Eq. 3.3. The EM algorithm is a method to find maximum likelihood estimates of parameters. The EM algorithm iteratively alternates between an expectation (E) and maximization (M) steps to maximize the expected likelihood using the estimated parameters. The following steps are iteratively applied until convergence:

E-step The expected assignment $\hat{z}_{ij}$ is computed based on the estimates $\hat{w}_j$ and $\hat{O}_j$ of the M-step.

$$\hat{z}_{ij}^{(t)} = \frac{\hat{w}_j^{(t)} P(F_i = \hat{O}_j^{(t)})}{\sum_{j=1}^{k} \hat{w}_j^{(t)} P(F_i = \hat{O}_j^{(t)})} \tag{3.4}$$

where t represents the t-th iteration. We initialize $\hat{z}_{ij}$ based on the partitioned dendrogram at a given height whether a read i is assigned to a cluster j or not, *i.e.*, $\hat{z}_{ij}^{(0)} \in \{0, 1\}$.

M-step The parameter $\hat{w}_j$ and $\hat{O}_j$ are calculated using the assignment $\hat{z}_{ij}$ from the previous E-step.

$$\hat{w}_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\hat{z}_{ij}^{(t)} \tag{3.5}$$

$$\hat{O}_j{}^{(t+1)} = \arg\min_{F\in\mathbf{F}}\left[\sum_{i=1}^{n}\hat{z}_{ij}^{(t)}d(F_i,F)\right] \tag{3.6}$$

We iterate these procedures using the base-called sequences $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ of the representative flowgrams $\hat{\mathbf{F}}$ by substituting $\mathbf{F}$ in each equation with $\mathbf{S}$.

3.2.3 Web Server Implementation

Figure 3.4(c) shows the internal architecture of the web server, which consists of a front-end for user interaction and a back-end for computation. A user uploads a compressed input data file, which can be demultiplexed and pre-filtered by preprocessing scripts provided on the web page. The user can also specify analysis options and additional information such as the barcode and primer used. The output includes denoised reads, a rarefaction curve showing the number of 3% OTUs versus the number of samples, a pie chart representing the diversity of OTUs estimated, and a multiple sequence alignment of the sequences in each OTU.

To implement the front-end, PHP with CodeIgniter and JavaScript with jQuery were used. For asynchronous data transfers from clients to the server, Ajax [92] was used. MUGAN uses Apache HTTP Server for HTTP service and MySQL Server for user data management. For visualization of each OTU, our web server carries out multiple sequence alignment using Clustal Omega [93] and employs MView [94] to visualize multiple sequence alignment results.

There are three major steps in the back-end analysis: removal of indel type pyrosequencing noise, removal of PCR noise appearing as substitutions, and estimation of OTUs by clustering denoised sequences. Our approach employs a CPU-GPU hybrid computing approach and executes the remaining back-end

steps on multi-core CPUs using shared-memory-based parallelization of the EM and hierarchical clustering algorithms used in the AmpliconNoise pipeline. For efficient job management and multi-user support, Gearman, an application framework that acts as a task queue was used. Multiple user requests are queued in this task queue, resolving the limitation that only one GPU kernel can be launched on a device at a time.

3.3 Experimental Results

3.3.1 Experiment Setup

We tested our approach using twelve public datasets of a mixture of full-length 16S rRNA clones, eight were obtained from Quince *et al.* [27] and the remaining four from the Human Microbiome Project Data Analysis and Coordination Center (HMP DACC). They were sequenced using a 454 GS FLX and 454 GS FLX Titanium pyrosequencer. The read lengths from the GS FLX and GS FLX Titanium are 250 and 500 bp, respectively. The detailed specification of the machine used for each experiment is given in the caption of each experimental result. The baseline for comparison was done using a dataset from Quince *et al.* [27] on the same machine with 8-node MPI cluster. For the experiment under a cloud computing environment, we used Amazon Web Services (AWS) with a 40-node cluster. Note that the running time presented in this section includes I/O time, unless otherwise stated.

3.3.2 Accuracy Comparison

To evaluate the accuracy of our approach, the number of estimated OTUs was compared to that of the baseline under the same circumstance (*i.e.*, machine and time). The estimated OTUs are the most commonly used unit in the microbiome analysis and defined on a specific sequence difference threshold on a dendrogram. The difference ratio of 0.03 and 0.05 are used as boundaries

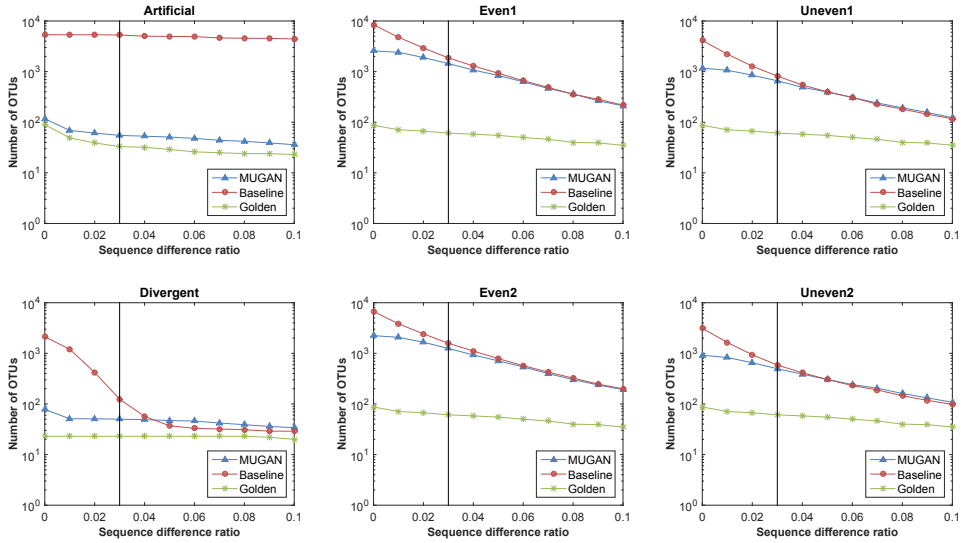
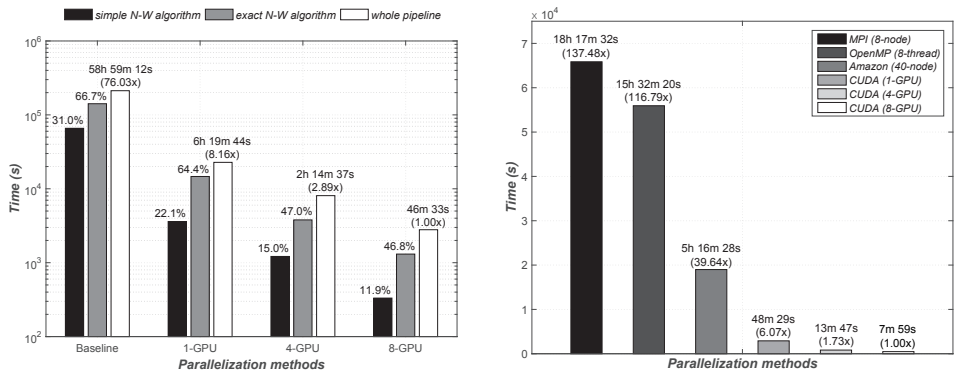


Figure 3.5: Comparison of the number of OTUs estimated by MUGAN and AmpliconNoise (baseline) given the same amount of runtime.

for defining species and genera, respectively. We evaluated the number of estimated OTUs on the threshold 0.03. It is known that the sequencing errors inflate the number of estimated OTUs [25]; hence, the method which derives an estimated OTU number similar to that of the golden reference has better performance. Note that using different thresholds derives different estimated OTU numbers.

Figure 3.5 shows the number of estimated OTUs derived by our approach and the baseline over sequence difference ratios from 0.00 to 0.10 using the six datasets sequenced by the GS FLX. In the difference ratio of 0.03, *i.e.*, the criteria to define species, our approach estimated 96.2 and 2.48 times fewer OTUs from the Artificial and Divergent datasets than the baseline, respectively.

In certain cases, the baseline appeared to have better performance than our approach had (*e.g.*, the sequence difference ratio of 0.05 or higher for the Divergent data), but this was due to an artifact of the experimental setup, which was to compare the performance obtained by different methods given the same amount of runtime. In the experiments for the accuracy compar-



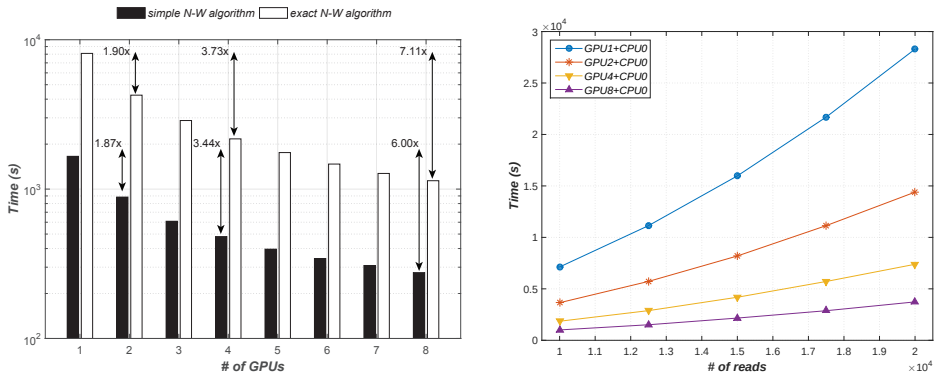
(a) (b)

Figure 3.6: Comparison of the different parallelization methods. (a) Whole pipeline with breakdowns. (b) Exact N-W algorithm only.

ison, the distance between a pair of sequences is initialized as infinity. The sequence pairs that are not yet aligned in the given time limit remain to have infinite distance values, and they are excluded from clustering, thus lowering the number of clusters found. Since the baseline software is significantly slower than our approach, it is possible that the number of OTUs estimated by the baseline seems to be smaller than our approach. However, if enough time is given, then such unclustered sequence pairs would disappear.

3.3.3 Running Time Comparison

Figure 3.6(a) shows the elapsed user time of our approach and the baseline, along with the two most time-consuming steps based on N-W algorithm as listed in Table 3.2. As shown in this figure, using one and eight GPUs are 9.34 and 76.03 times faster than the baseline, respectively. The performance did not linearly increase due to the rate-limiting steps other than the N-W algorithm-based steps that do not need additional parallelization. Figure 3.6(b) shows the elapsed user time of the exact N-W algorithm-based step with regard to different parallel computing platforms. Among the four parallel computing platforms, our multi-GPU-based approach outperformed the alternatives by



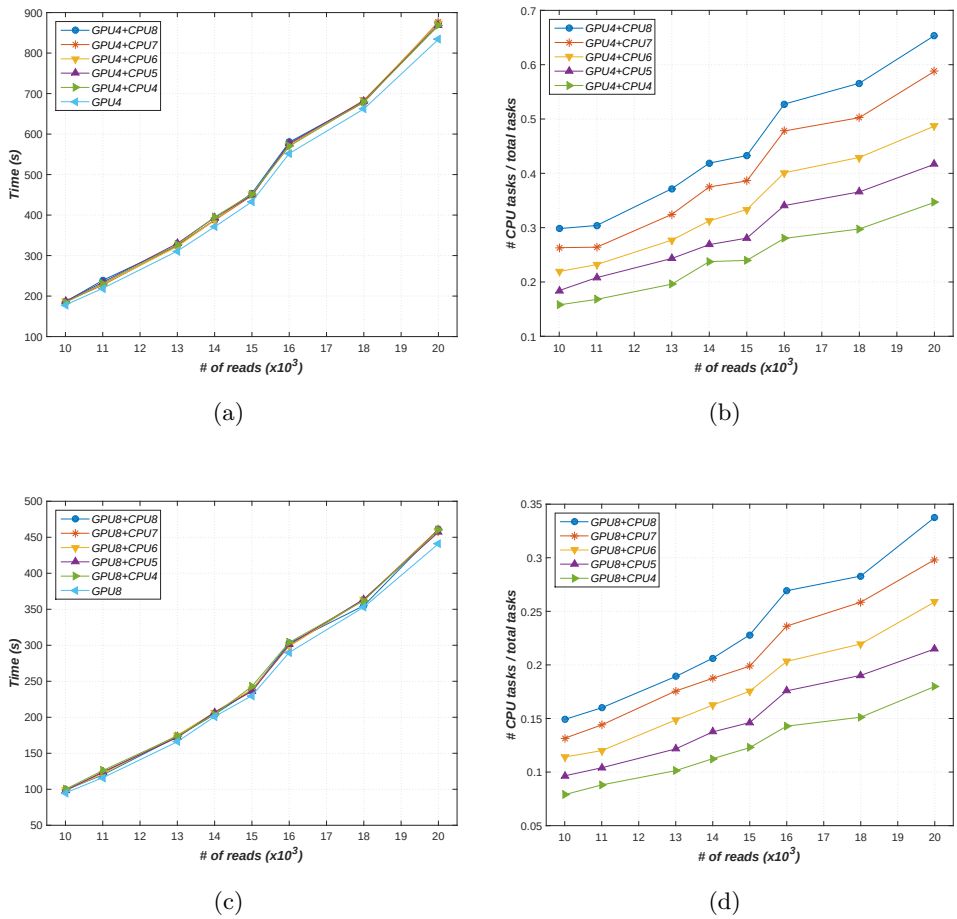
(a) (b)

Figure 3.7: Comparison of the scalability on the performance. (a) Different number of GPUs. (b) Different number of reads.

a large margin. Under the cloud computing circumstance on AWS, it took more than the estimated time based on the scalability analysis. This might be caused by the latency in AWS that was not considered here and we would need over 100 nodes on AWS to yield the performance similar to that of one GPU.

Figure 3.7(a) shows the effect of the number of used GPUs on the elapsed user time. The performance improvement was nearly proportional to the number of GPUs used. Hence, using more GPUs will further increase the maximum performance, which is reported in this paper (76.03 times over the baseline). Figure 3.7(b) shows the elapsed user time of the exact N-W algorithm-based step with regard to the various numbers of reads. While considering the complexity of the N-W algorithm, our approach maintains its scalability as the number of reads and used GPUs increases.

Figure 3.8 shows the variation in performance depending on implementation of multi-GPU and CPU co-processing. In this figure, as the number of CPUs in computation increases, the portion of submatrices that are distributed to CPUs also increases as is observed in Figure 3.8(b) and 3.8(d). Hence, utilizing CPUs not only for managing GPUs but also for computation



(a) (b)

(c) (d)

Figure 3.8: Comparison of the hybrid settings. (a) Runtime of 4-GPU. (b) Scheduling of 4-GPU. (c) Runtime of 8-GPU. (d) Scheduling of 8-GPU.

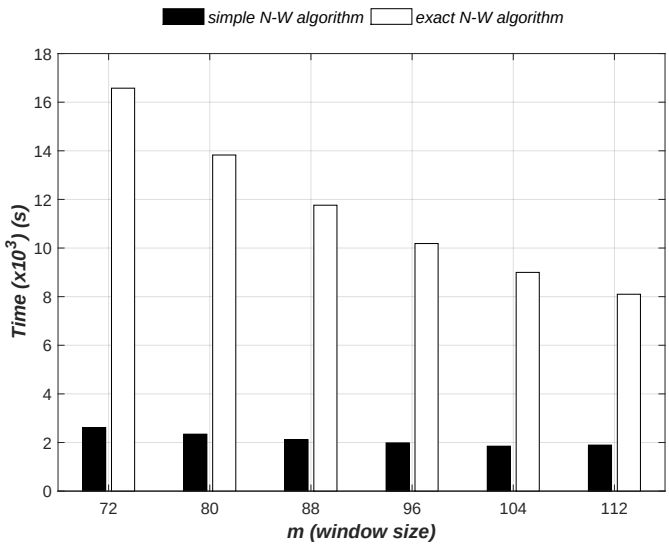


Figure 3.9: Comparison of the window size m on the performance.

did not give a significantly better result than using CPUs only for managing GPUs as is observed in Figure 3.8(a) and 3.8(c).

Figure 3.9 shows the effect of the submatrix size on the performance. As the size of submatrix was increased from 72 to 112, the elapsed user time decreases as there are enough threads to form a new block that can start processing in parallel.

Table 3.3 lists the elapsed user time of our approach for processing each of the twelve datasets. For comparison, the table also shows the elapsed user time of the baseline executed on our web server with GPU parallelization disabled. Overall, our approach delivers substantially improved runtime over the baseline, demonstrating the effectiveness of our multi-GPU acceleration. For the GS FLX datasets, our approach runs 2.2–36.1 times faster than the baseline, whereas for the GS FLX Titanium datasets, it runs 11–151 times faster. That is, the amount of speedup by our approach is more noticeable for the GS FLX Titanium datasets in which individual reads are longer than those in the GS FLX datasets. Our approach leverages this increased efficiency to achieve a greater accuracy than the baseline in the estimation of the number

Table 3.3: Runtime of MUGAN for various datasets

dataset	# reads	runtime (MUGAN)			baseline
		total	GPU	CPU	(w/o GPU)
Artificial†	31,867	19m 50s	14m 26s	5m 24s	81m 34s
Divergent†	35,190	31m 30s	7m 22s	24m 08s	125m 41s
Even1†	53,771	42m 31s	13m 40s	28m 51s	$\sim$ 1 week
Even2†	45,178	43m 24s	8m 51s	34m 33s	171m 32s
Even3†	54,153	39m 52s	11m 33s	28m 19s	$\sim$ 1 week
Uneven1†	44,926	47m 37s	5m 37s	42m 00s	102m 47s
Uneven2†	44,176	28m 41s	4m 37s	24m 04s	82m 40s
Titan†	25,438	145m 56s	95m 29s	50m 27s	1742m 48s
SRR072218‡	101,022	413m 54s	218m 57s	199m 57s	$\sim$1 month
SRR072219‡	92,417	328m 50s	148m 53s	179m 57s	$\sim$1 month
SRR072222‡	94,333	285m 59s	135m 41s	150m 18s	$\sim$1 month
SRR072223‡	94,698	288m 48s	145m 16s	143m 32s	$\sim$1 month

Source: †Quince *et al.* [27], ‡HMP DACC (http://www.hmpdacc.org). [CPU: Intel Xeon X5650 @ 2.67GHz $\times$ 2, RAM: 64-GB DDR3, GPU: NVIDIA GeForce GTX 580 (3-GB GDDR5) $\times$ 4]

of OTUs in a sample, given the same amount of time (Figure 3.5).

To summarize, our approach significantly reduces the time demand for estimating the microbial community diversity and provides intuitive visualization of results.

3.3.4 Understanding Output

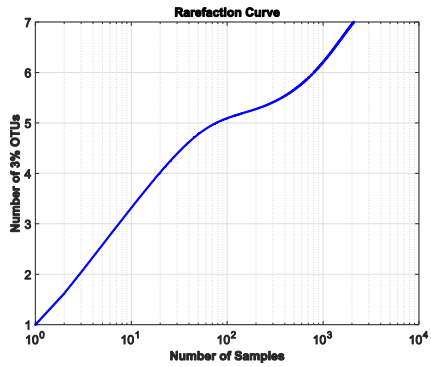
The user retrieves the analysis results using the URL provided by our web server or from a notification email. The output from our web server includes the following:

- A table listing some basic statistics on the result (Figure 3.10(a)).
- A rarefaction curve showing the number of species as a function of the number of samples (Figure 3.10(b)). Rarefaction is a technique used to assess the richness of species from the results of sampling.
- A pie chart representing the composition of different OTUs (Figure 3.10(c)). Hovering the mouse cursor over the area of an OTU will show the ID and the number of sequences included in the OTU. Clicking the area of

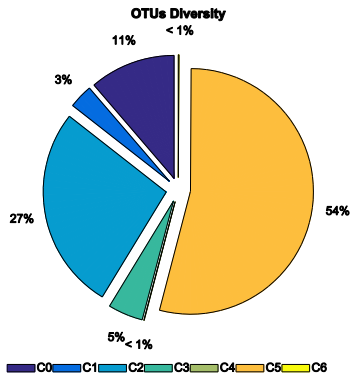
A

Sample Name	Input Reads	Clean Sequences	Clean Reads
C005	2094	16	2094

B



C



D

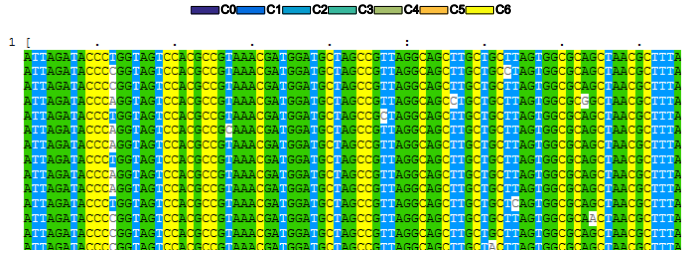


Figure 3.10: Outputs from MUGAN: (a) Basic statistics of the result. (b) A rarefaction curve showing the number of species as a function of the number of samples. (c) A pie chart representing the composition of different OTUs. (d) A multiple sequence alignment of the sequences included in each OTU.

an OTU will show the multiple alignment result of the sequences that belong to the OTU.

- A multiple sequence alignment of the sequences included in each OTU (Figure 3.10(d)).

3.4 Discussion

Owing to an increasing importance of the role of human microbiome in health and disease, developing methods to read and analyze the overwhelming amount of sequence data in a robust and efficient manner is essential. According to our experimental results, our approach significantly outperforms the original AmpliconNoise in terms of accuracy, runtime, and scalability. As stated in Section 3.1.4, the pairwise distance calculation between sequences occupies about 97% of the total runtime, while the remaining 3% is derived mainly from the EM algorithm for Gaussian mixture models. This observation justifies our choice of GPU-based parallelization of the pairwise distance calculation. We could further accelerate the EM algorithm with CUDA, given that EM can also be made significantly faster than a naïve single threaded implementation [95]. A future revision of our approach may thus include this parallelization for additional speed-up.

GPU-accelerated clusters provide the tremendous computing power required by computationally intensive workloads as stated in Section 3.3. Nonetheless, the data transfer overhead between the device and the host diminishes overall performance as shown in Figure 3.7(a). Our approach launches two kernels that transfer backtracking information and aligned sequences between the device and the host for each $m \times m$ window. These transfers lower the performance of our approach as they cause idling threads before the next windows are launched. By pipelining the sequence alignment computation and the data transfer of the aligned sequences, we will be able to reduce data transfer

overhead. We consider this task as a part of our future work. Additionally, to resolve the degradation in performance due to idling threads, we are planning to use new system architectures (such as GPUDirect [96] or hUMA [97]).

Pyrosequencing is widely employed for microbiome analysis using 16S rRNA as it has advantages for metagenomic analysis based on unassembled reads [98, 99]. As 16S rDNA Illumina tags are known to provide more realistic estimates of community richness and evenness than amplicon tags, Illumina sequencing is emerging as an alternative for probing microbial diversity [100]. To prepare for this technological change, it will be important to develop an embracing algorithm for sequencing error-correction as different sequencing technologies have different error characteristics and profiles [10], although there are common components and premises. Therefore, our future focus includes revising our approach such that it can handle both pyrosequenced amplicon sequences and those produced by Illumina sequencing.

3.5 Summary

Metagenomic sequencing has become a crucial tool for obtaining a gene catalogue of OTUs in a microbial community. A typical metagenomic sequencing produces a large amount of data (often in the order of terabytes or more), and computational tools are indispensable for efficient processing. In particular, error correction in metagenomics is crucial for accurate and robust genetic cataloging of microbial communities. However, many existing error-correction tools take a prohibitively long time and often bottleneck the whole analysis pipeline. To overcome this computational hurdle, we analyzed and exploited the data-level parallelism that exists in the error-correction procedure and proposed a tool named MUGAN that exploits both multi-core CPUs and GPUs for co-processing. According to the experimental results, our approach reduced not only the time demand for denoising amplicons, but also the overestimation of the number of OTUs.

Chapter 4

Sequence Interaction Learning

MicroRNAs (miRNAs) are small non-coding RNA molecules that can control the function of their target messenger RNAs (mRNAs) by down-regulating the expression of the targets [101]. By controlling the gene expression at the RNA level, miRNAs are known to be involved in various biological processes and diseases [102]. As miRNAs play a central role in the post-transcriptional regulation of more than 60% of protein coding genes [32], investigating miRNAs is of utmost importance in many disciplines of life science. As explained further in Section 4.1.3, miRNAs are derived from the precursor miRNAs (pre-miRNAs) and then exhibit their regulatory function by binding to the target sites present in mRNAs. Two types of computational problems about miRNAs thus naturally arise in bioinformatics: miRNA host identification (*i.e.*, the problem of locating the genes that encode pre-miRNAs) and miRNA target prediction (*i.e.*, the task of finding the mRNA targets of given miRNAs). This chapter focuses on the target prediction problem.

This chapter proposes deepTarget, an end-to-end machine learning framework for robust miRNA target prediction. deepTarget adopts deep recurrent neural network (RNN)-based auto-encoders to discover the inherent represen-

tations of miRNA and mRNA sequences and utilizes stacked RNNs to learn the sequence-to-sequence interactions underlying miRNA-mRNA pairs. Leveraged by this combination of unsupervised and supervised learning approaches, deepTarget not only delivers an unprecedented level of accuracy but also eliminates the need for manual feature extraction. Furthermore, according to our visual inspection of the activation of the RNN layers used in deepTarget, meaningful patterns appeared in the nucleotide positions that corresponded to the real miRNA-mRNA binding sites. Further analyzing the activation patterns may allow us to obtain novel biological insight into regulatory interactions.

4.1 Background

miRNAs play a crucial role in controlling the function of their target genes by down-regulating gene expression, actively participating in various biological processes. To predict targets of given miRNAs, numerous computational tools have been proposed [34, 103].

There exists a recent study [104] that used convolutional neural network (CNN) with 20 manually crafted features to predict the target of a miRNA. In [104], however, the deep learning method was used as a classifier rather than a feature learner from the raw features (*i.e.*, biological sequences). The novelty of our approach comes from its use of deep RNNs without laborious feature engineering: deepTarget performs RNA sequence modeling using autoencoders and learns the interactions between miRNAs and mRNAs using stacked RNNs. To the best of our knowledge, the proposed methodology is one of the first attempts to predict miRNA targets by sequence modeling using deep RNNs.

4.1.1 Autoencoder

An autoencoder is an artificial neural network used for learning representations. Autoencoders have almost similar structure to that of multilayer perceptrons (MLPs) except that the output layer has the same number of nodes

as the input layer. The objective of an autoencoder is to learn a meaningful encoding for a set of data in an unsupervised manner while a MLP is to be trained to predict the target value in a supervised manner.

Typically, an autoencoder consists of two parts, an encoder and a decoder, which can be defined as transitions ϕ and ψ, respectively, given by

$$\phi : \mathcal{S} \mapsto \mathcal{X} \quad \text{and} \quad \psi : \mathcal{X} \mapsto \mathcal{S} \tag{4.1}$$

subject to

$$\arg\min_{\phi,\psi} \|S - (\psi \circ \phi)S\|^2 \tag{4.2}$$

for $S \in \mathcal{S}$, where $\mathcal{S}$ is the input space and $\mathcal{X}$ is the mapped feature space.

Various techniques exist to improve the generalization performance of autoencoders. Examples include the denoising autoencoder (DAE) [105], and the variational autoencoder (VAE) [106]. Recently, the VAE concept has become more widely used for learning generative models of data [107].

In this chapter, RNN-based autoencoders are used to model miRNA and mRNA sequences to extract their inherent features in an unsupervised manner.

4.1.2 Recurrent Neural Network (RNN)

An RNN is an artificial neural network where connections between units form a directed cycle. This creates an internal state of the network that allows it to exhibit dynamic temporal behavior. RNNs can use their internal memory to process sequences of inputs. This makes RNNs applicable to various tasks, such as unsegmented connected handwriting recognition [108] and speech recognition [109].

Leveraged by advances in the memory-based units [*e.g.*, long short-term memory (LSTM) [110] and gated recurrent unit (GRU) [111]], network architecture [*e.g.*, stacked RNNs [112] and bidirectional RNNs [113]] and training

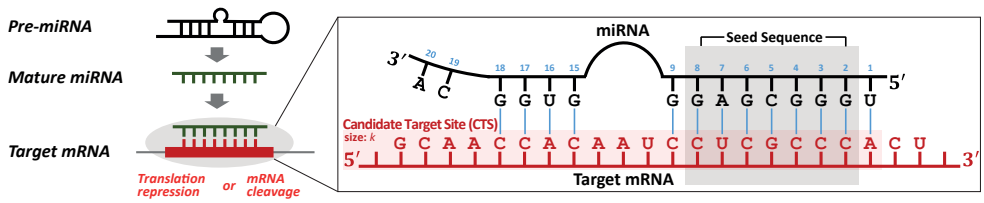


Figure 4.1: miRNA biogenesis and its function.

methods [*e.g.*, iRNN [108] and dropout [114]], RNNs are delivering breakthrough performance in tasks involving sequential modeling and prediction [115, 116, 109, 117].

According to Goodfellow *et al.* [118], RNNs can do various sequence modeling tasks including the following:

T1: mapping an input sequence to a fixed-size prediction

T2: modeling a distribution over sequences and generating new ones from the estimated distribution

T3: mapping a fixed-length vector into a distribution over sequences

T4: mapping a variable-length input sequence to an output sequence of the same length

T5: mapping an input sequence to an output sequence that is not necessarily of the same length

In this chapter, RNNs are used to perform task **T5** for sequence modeling using autoencoders and task **T1** for target prediction.

4.1.3 Biology of miRNA-mRNA Interactions

In molecular biology, directionality of a nucleic acid, 5'-end and 3'-end, is the end-to-end chemical orientation of a single strand of the nucleic acid. The 3' untranslated region (UTR) of mRNA is the region that directly follows the translation termination codon. As shown in Figure 4.1, miRNAs exhibit

their function by binding to the target sites present in the 3' UTR of their cognate mRNAs. By binding to target sites within the 3' UTR, miRNAs can decrease gene expression of mRNAs by either inhibiting translation or causing degradation of the transcript. The *seed* sequence of a miRNA is defined as the first two to eight nucleotides starting from the 5' to the 3' UTR. The *candidate target site* (CTS) refers to the small segment of length k of mRNA that contains a complementary match to the seed region at the head. The blue line between a pair of nucleotides (one from miRNA and the other from mRNA) in Figure 4.1 represents the Watson-Crick (WC) pairing formed by sequence complementarity (*i.e.*, the binding of `A-U` and that of `C-G`).

The canonical sites are complementary to the miRNA seed region while the non-canonical sites are with bulges or single-nucleotide loops in the seed region. The non-canonical sites include two types, 3' compensatory and centered sites. The proposed machine learning-based approach has potential to learn both canonical and non-canonical sites. While considering CTSs to predict targets of given miRNAs, there are four types of seed sequence matches [34] often referred to as 6mer, 7mer-m8, 7mer-A1, and 8mer in the literature. The details of each type are as follows:

- 6mer: exact WC pairing between the miRNA seed and mRNA for six nucleotides
- 7mer-m8: exact WC pairing from nucleotides 2–8 of the miRNA seed
- 7mer-A1: exact WC pairing from nucleotides 2–7 of the miRNA seed in addition to an `A` of the miRNA nucleotide 1
- 8mer: exact WC pairing from nucleotides 2–8 of the miRNA seed in addition to an `A` of the miRNA nucleotide 1

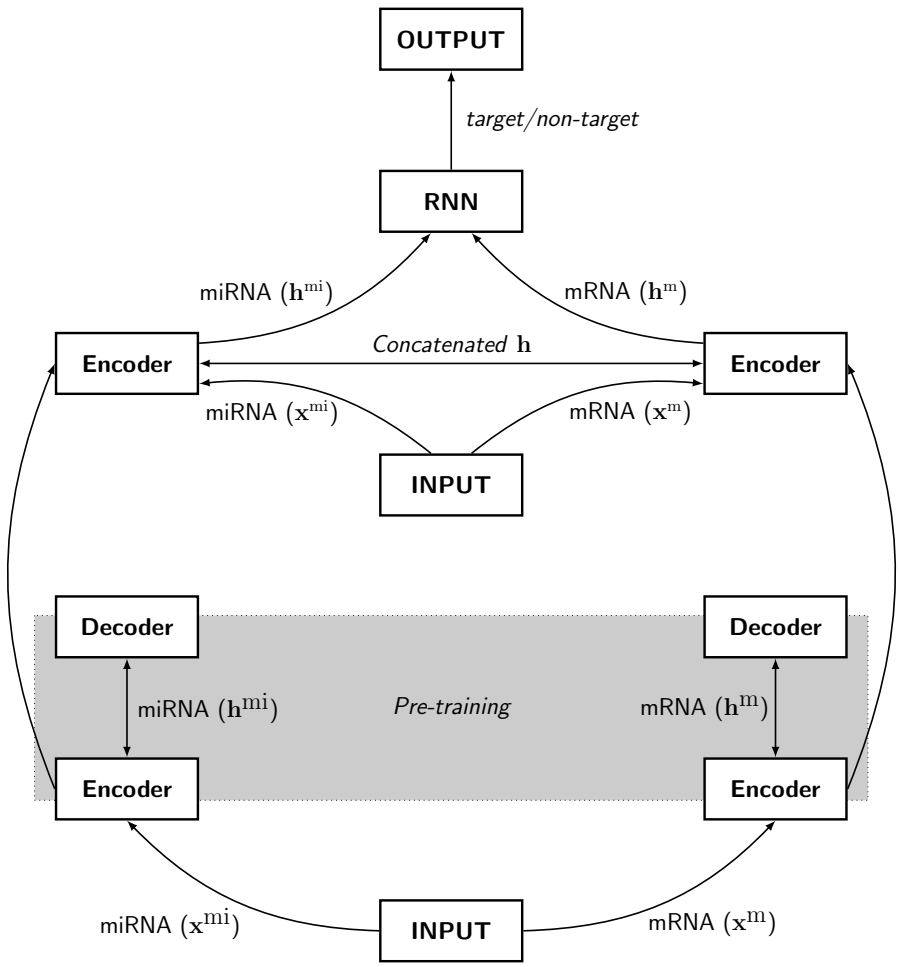


Figure 4.2: Overview of proposed deepTarget methodology.

4.2 Methods

Figure 4.2 shows the overview of the proposed deepTarget methodology. Our method is based on RNA sequence modeling and sequence-to-sequence interaction learning using autoencoders and stacked RNNs. The input layer is connected to the first layer of two autoencoders to model miRNA and mRNA sequences, respectively. The second layer is an RNN layer to model the interaction between miRNA and mRNA sequences. The outputs of the top RNN layer are fed into a fully connected output layer, which contains two units for classifying targets and non-targets.

We preprocess the CTSs with length k, which are empirically determined by all of the four seed matching types between miRNAs and mRNAs as described in Section 4.1.3. After preprocessing, as shown in Algorithm 3, our approach proceeds in two major steps with miRNA and CTS sequence pair as an input: (1) unsupervised learning of two autoencoders for modeling miRNAs and mRNAs (lines 4–7), and (2) supervised learning of the whole architecture for modeling interaction between miRNAs and mRNAs (lines 10–12).

Throughout this paper, we denote a sequence as $\mathbf{s}^{\text{mi}}$ of n_d^{mi}-dimensional vector for miRNA and $\mathbf{s}^{\text{m}}$ of n_d^{m}-dimensional vector for mRNA where n_d^{mi} and n_d^{m} can, in general, be different and $k \approx \max(n_d^{\text{mi}})$.

4.2.1 Input Representation

Each RNA read is a sequence that has four types of nucleotides $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$ and needs to be converted into numerical representations for machine learning.

A widely used conversion technique is using one-hot encoding [119], which converts the nucleotide in each position of a DNA sequence of length n_d into a four-dimensional binary vector and then concatenates each of the n_d four-dimensional vectors into a $4n_d$-dimensional vector representing the whole sequence. For instance, let $s \in \mathcal{S}$ where $\mathcal{S} = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$, then, a sequence $\mathbf{s} = (\texttt{A}, \texttt{G}, \texttt{U}, \texttt{U})$ where $n_d = 4$ is encoded into a tuple of four 4-dim binary

Algorithm 3 Pseudo-code of deepTarget

Input: N encoded RNA sequences, $\mathbf{x}_1^{\text{mi}}, \cdots, \mathbf{x}_N^{\text{mi}}$ and $\mathbf{x}_1^{\text{m}}, \cdots, \mathbf{x}_N^{\text{m}}$
Output: $\mathbf{y}$ (target/non-target)
Pre-train the autoencoders AE^{mi} and AE^{m} ▷ AE^{mi}, AE^{m}: the autoencoders described in Section 4.2.2
repeat
    minimize the reconstruction error $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$
until # of epoch is n_{epoch}
Define the whole architecture [$\text{EN}^{\text{mi}} \parallel \text{EN}^{\text{m}}$]-RNN ▷ EN^{mi}, EN^{m}: the encoders in described in Section 4.2.3
Concatenate two representations, $\mathbf{h}^{\text{mi}}$ and $\mathbf{h}^{\text{m}}$, into $\mathbf{h}$ ▷ $\mathbf{h}^{\text{mi}}$, $\mathbf{h}^{\text{m}}$: the outputs of each encoder
Fine-tune the architecture
repeat
    minimize the logarithmic loss $\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$
until # of epoch is n_{epoch}

vectors:

$$\langle [1,0,0,0], [0,0,1,0], [0,0,0,1], [0,0,0,1] \rangle.$$

According to recent studies [120, 121], however, applying the vanilla one-hot encoding scheme tends to give limited generalization performance caused by the sparsity of the encoding. In lieu of the one-hot encoding scheme, our approach thus encodes each nucleotide into a four-dimensional dense vector that is randomly initialized and trained by the gradient descent method through the whole architecture as a normal neural network layer [122]. We encode sequences to $\mathbf{x}^{\text{mi}}$ for miRNA and $\mathbf{x}^{\text{m}}$ for mRNA, where $\mathbf{x}$ is a tuple of n_d four-dimensional dense vectors. For instance, a one-hot encoded tuple $\langle [0,0,0,1], [0,0,1,0] \rangle$ is encoded into a dense tuple

$$\langle [-0.1, -0.2, 0.1, 0.3], [-0.1, -0.2, 0.3, 0.4] \rangle.$$

4.2.2 Modeling RNAs using RNN based Autoencoder

Menor *et al.* [36] described 151 site-level features between miRNA-target pairs for target prediction, categorizing them into seven groups: binding energy, the type of seed matching, miRNA pairing, target site accessibility, target site composition, target site conservation, and the location of target sites. To relieve these feature engineering required for target prediction in conventional approaches, deepTarget exploits the unsupervised feature learning using the RNN encoder-decoder model [123].

Each of our autoencoder model consists of two RNNs, one as an encoder and the other as a decoder. The encoder RNN encodes $\mathbf{x}$ to $\mathbf{h}$, and the following decoder RNN decodes $\mathbf{h}$ to reconstructed $\hat{\mathbf{x}}$, which minimizes the reconstruction error

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \tag{4.3}$$

where $\mathbf{h}$ is a tuple of n_d n_h-dimensional vectors (here n_h is the number of the hidden units of the encoder). Through unsupervised learning of these autoencoders, we get the representations of inherent features that will be used by the stacked RNN layer described in Section 4.2.3.

After unsupervised learning of these two autoencoders for miRNA and mRNA, respectively, we bypass the decoder and connect the encoder directly to the next layer as a tuple of the fixed dimension n_h representations as shown in Figure **??**. This restriction of the fixed dimension n_h is to model the activations of the second layer to be analogous to target pairing patterns. We then obtain sequence representations $\mathbf{h}^{\mathrm{mi}}$ for miRNA and $\mathbf{h}^{\mathrm{m}}$ for mRNA.

4.2.3 Modeling Interaction between RNAs

The miRNA-mRNA binding sites can be classified into three types [124]: 5'-dominant canonical, 5'-dominant seed only, and 3'-compensatory. Regardless of the type of a binding site, its detection typically requires the sequence

alignment procedure in most of the conventional approaches. However, sequence alignment may often limit the robustness of target prediction because of the need to tune various alignment parameters. The use of an RNN-based approach allows us to omit the sequence alignment procedure (Refer to Figure 4.6 to see that our RNN-based approach can still detect the binding sites without any alignment).

In this interaction modeling step, the learned features of miRNAs $\mathbf{h}^{\mathrm{mi}}$ and mRNAs $\mathbf{h}^{\mathrm{m}}$ (extracted by the two autoencoders described in Section 4.2.2) need to be combined into one tuple $\mathbf{h}$. To this end, we can devise various ways to merge the representations between each model (*e.g.*, summation, multiplication, concatenation, and average).

The concatenated representations from different modalities have been unsuccessful in constructing a high-dimensional vector since the correlation between features in each modality is stronger than that between modalities [125]. To resolve this issue, deep learning methods have been proposed to learn joint representations that are shared across multiple modalities after learning modality-specific network layers [126]. In this paper, we adopt the method of concatenating each dimension, $\mathbf{h}^{\mathrm{mi}}$ and $\mathbf{h}^{\mathrm{m}}$, into tuple $\mathbf{h}$ given by

$$\mathbf{h} = \langle \mathbf{h}_1, \cdots, \mathbf{h}_{n_d} \rangle = \langle (\mathbf{h}_1^{\mathrm{mi}}, \mathbf{h}_1^{\mathrm{m}}), \cdots, (\mathbf{h}_{n_d}^{\mathrm{mi}}, \mathbf{h}_{n_d}^{\mathrm{m}}) \rangle. \tag{4.4}$$

The two unsupervised-learned encoders followed by a stacked RNN layer comprises our proposed architecture for target prediction that has $\mathbf{x}^{\mathrm{mi}}$ and $\mathbf{x}^{\mathrm{m}}$ as the inputs. The output node has an activation probability given by

$$P(Y = i|\mathbf{h}) = \frac{1/(1 + \exp(-\mathbf{w}_i^T \mathbf{h}))}{\sum_{k=0}^{1} 1/(1 + \exp(-\mathbf{w}_k^T \mathbf{h}))} \tag{4.5}$$

where y is the label whether the given pair is a miRNA-target pair ($y = 1$) or not ($y = 0$), and $k \in \{0, 1\}$ is the class index. The objective function $\mathcal{L}(\mathbf{w})$

that has to be minimized is then as follows:

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i)) \tag{4.6}$$

where p_i is the probability of the given pair $(\mathbf{x}^{\mathrm{mi}}, \mathbf{x}^{\mathrm{m}})$ being a valid miRNA-target pair and N is the mini-batch size used.

4.3 Experimental Results

4.3.1 Experiment Setup

Dataset

In the literature, there exist two types of datasets that contain miRNA-mRNA pairing information: *site-level* and *gene-level* datasets. As depicted in Figure 4.3, let us assume that there are four CTSs on a gene sequence, two of which are true target sites of a miRNA. A site-level dataset has a label for each CTS indicating whether this CTS is a target or not, and we can evaluate the performance of a target prediction tool for each CTS. On the other hand, a gene-level dataset has a label for each gene, not for each CTS, and we can evaluate the prediction performance at the gene level. Refer to the caption for Figure 4.3 for additional details.

To test the proposed deepTarget, we utilized a public human miRNA-mRNA pairing data repository [36], which provides both site-level and gene-level datasets. The creators of this repository obtained validated target sites from miRecords [127] database and mature human miRNA sequences from mirBase [128] database. We used 2,042 human miRNAs that are known to bind to their cognate mRNAs. To train deepTarget, we utilized the 507 site-level and 2,891 gene-level miRNA-target site pairs as the positive training set. The negative training set was generated as explained in the next subsection.

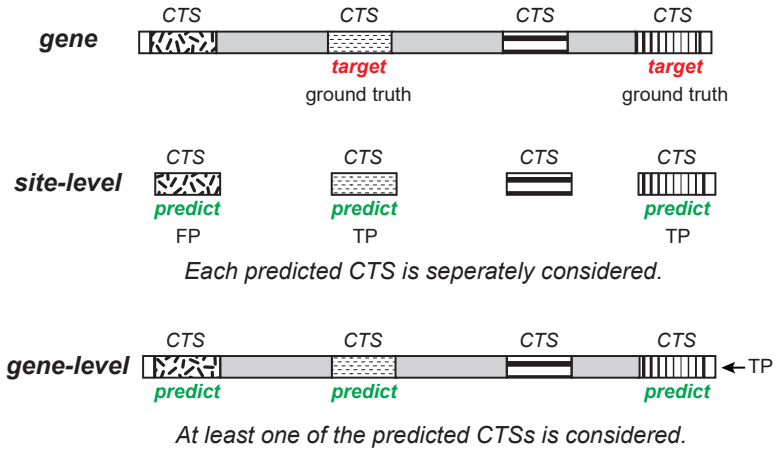


Figure 4.3: Definition of site-level and gene-level target pair datasets.

Negative Training Data Generation

The negative dataset was generated using mock miRNAs in a similar manner to [129, 130] that did not have overlap with the seed sequences of any real mature miRNAs in mirBase. The mock miRNAs were generated with random permutaions using the Fisher-Yates shuffle algorithm [131]. The 507 for site-level and 3,133 for gene-level negative mock miRNA-target pairs were then generated for each real miRNA-target pair in the positive dataset by replacing the positive miRNA in the real miRNA-target pair. The negative target regions were generated for each mock miRNA-target pair using MiRanda [132] by thresholding the minimum alignment score for minimizing the biases relevant to the miRanda algorithm.

Note that the negative data generation procedure described above is widely used in the literature to handle lack of biologically meaningful and experimentally verified negative pairs for training. For example, the mock miRNA-based negative dataset we used is essentially identical to that used in [129, 130, 36]. In addition, as mentioned in [36], the mock miRNA-based negative dataset in lieu of real biological negative ones was chosen to obtain a balanced training dataset. These authors investigated four combinations of training and test

datasets with the real and mock sequences and found that the mock miRNAs for both model building and validation had the best predictive performance.

Network Architecure and Training

In the following sections, the architecture of our method is denoted by the number of nodes in each layer. For example, an architecture 4-60-30-2 has four layers with four input units, 60 units in the first hidden layer, and so on.

The proposed RNN-based approach used [(4-30-4) ‖ (4-30-4)]-(60-30)-2 where (4-30-4) is for the autoencoder described in Section 4.2.2 and (60-30) is for the stacked RNN described in Secion 4.2.3. Note that the symbol '‖' represents the two parallel autoencoders. The number of output units of each encoder for miRNA and mRNA, respectively, was set to an equal number in order to model the activations of the second layer to be analogous to target pairing patterns.

For training, our approach optimized the logarithmic loss function using Adam [133] [batch size: 50, the number of epochs: 50 (pre-training autoencoder), and 400 (fine-tuning architecture)]. The weights were initialized according to a uniform distribution as directed in [134]. We used dropout as the regularizer in lieu of recently popularized batch normalization [135] with 1,014 site-level target pairs.

The effects of varying architectures and dropout parameters will be presented in Section 4.3.

4.3.2 Prediction Performance

To evaluate the prediction performance of our deepTarget approach, we compared its performance with six existing tools for miRNA target prediction: MBSTAR [136], miRanda, PITA [137], RNA22 [138], TargetScan [139], and TargetSpy [140] with 6,024 gene-level target pairs. After we found CTSs in 6,024 gene-level target pairs, there remained 4,735 positive and 1,225 nega-

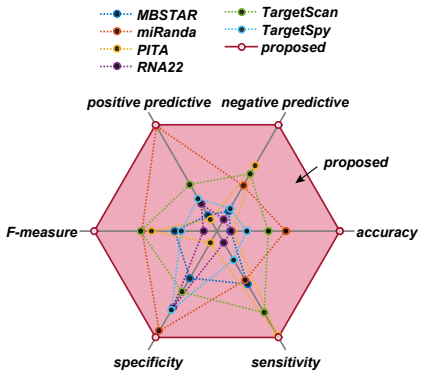


Figure 4.4: Comparison of prediction performance of proposed deepTartet and alternatives (best viewed in color).

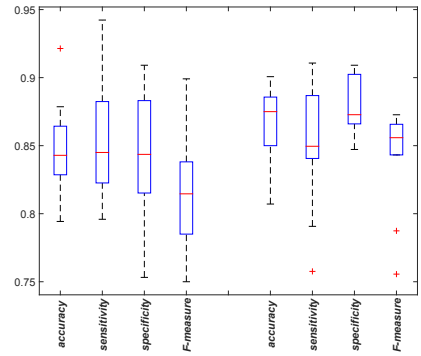


Figure 4.5: Test accuracy of our approach using two types of memory units (left: deep RNN with LSTM units, right: with GRU units).

tive pairs because of the disappearance of the seed in the mock miRNAs that match the mRNA sequences.

As a performance metric, we used the widely used measures including accuracy[1], sensitivity[2], specificity[3], F-measure[4], positive predictive value (PPV)[5], and negative predictive value (NPV)[6].

According to the definition of these metrics, they generally have FN in the denominator. Namely, high sensitivity and NPV represent that FN is low; however, the algorithms that classify most of the samples as positive would also have low FN. To distinguish such poor algorithms from more accurate ones, we also have to consider the specificity and PPV metrics, whose definitions include FP in the numerator.

Figure 4.4 and Table 4.1 show the comparison of prediction performance. To obtain the performance of deepTarget, we performed 10-fold cross validation and averaged each resulting value. Overall, our method significantly out-

[1]accuracy $= (TP+TN)/(TP+TN+FP+FN)$, where TP, FP, FN, and TN represent the numbers of true positives, false positives, false negatives, and true negatives, respectively.
[2]sensitivity $= TP/(TP+FN)$
[3]specificity $= TN/(TN+FP)$
[4]F-measure $= 2TP/(2TP+FP+FN)$
[5]PPV $= TP/(TP+FP)$
[6]NPV $= TN/(TN+FN)$

Table 4.1: Comparison of prediction performance of proposed deepTarget and alternatives

	MBSTAR	miRanda	PITA	RNA22	TargetScan	TargetSpy	deepTarget
accuracy	0.5805	0.7737	0.5870	0.5820	0.7125	0.6364	**0.9641**
sensitivity	0.5308	0.6122	0.9301	0.4006	0.7980	0.4934	**0.9389**
specificity	0.5308	0.9228	0.2703	0.7494	0.6336	0.7657	**0.9706**
F-measure	0.5921	0.7220	0.6837	0.4791	0.7271	0.5672	**0.9105**
PPV	0.5551	0.8797	0.5405	0.5960	0.6677	0.6616	**0.8848**
NPV	0.6114	0.7206	0.8074	0.5753	0.7727	0.6223	**0.9845**

Table 4.2: Effects of architecture variation on prediction performance

Architecture	1-layer	2-layer	3-layer	bidirectional
accuracy	0.8214	**0.8286**	0.8250	0.8250
sensitivity	**0.8429**	0.8351	0.8417	0.8254
specificity	0.8204	**0.8428**	0.8297	0.8041
F-measure	0.7745	**0.7838**	0.7759	0.7620

Table 4.3: Effects of changing dropout probability on prediction performance

Dropout	0.0	0.1	0.2	0.3	0.4
accuracy	0.8214	**0.8429**	0.8179	0.8179	0.7857
sensitivity	0.8429	**0.8480**	0.8460	0.8460	0.7980
specificity	0.8204	**0.8486**	0.8111	0.8111	0.7746
F-measure	0.7745	**0.7978**	0.7592	0.7592	0.7386

performed the best alternative in terms of accuracy, sensitivity, and F-measure by 24.61%, 17.66%, and 25%, respectively. As will be discussed in Section 4.4, biologists tend to give more priority to sensitivity than to specificity, when searching for potential targets of specific miRNA [141].

4.3.3 Effects of Architecture Variation

Figure 4.5 shows the effects of the memory unit used in deepTarget on its prediction performance. To generate this plot, we performed 10-fold cross validation and tested two popular memory units, LSTM and GRU. According to Jozefowicz *et al.* [142], it is believed that there is no architecture that can consistently beat the LSTM and the GRU in all conditions.

As shown in Figure 4.5, the architecture composed of GRU outperforms that of LSTM in our experiments. More quantitatively, the architecture using GRU as its memory unit showed 3.81%, 0.53%, 3.46%, and 5.06% improvements over the LSTM-installed architecture in terms of accuracy, sensitivity, specificity, and F-measure, respectively (the median values achieved were 0.8750, 0.8496, 0.8728, and 0.8559, respectively, in 10-fold cross validation).

Table 4.2 lists the effect of varying the architecture of the second layer RNN

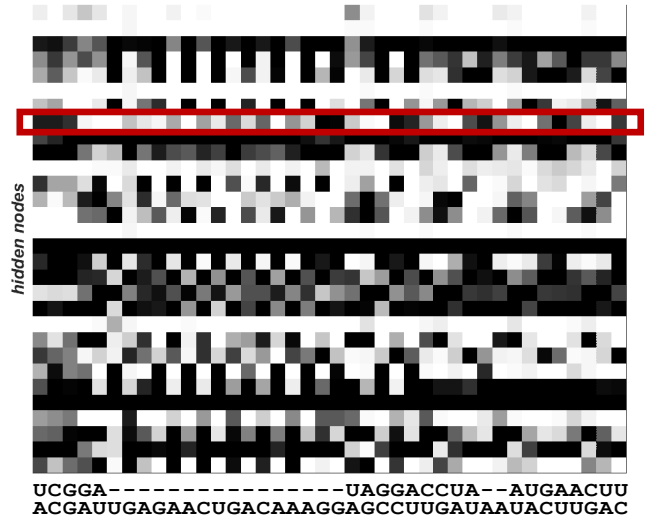


Figure 4.6: Change of activation of each node in the RNN layer over RNA sequences.

on overall performance. We tested stacked RNNs with one to three layers and single-layer bidirectional RNNs. Using the two layer architecture gave the best overall results, although the differences in performance were not significant.

Table 4.3 presents the effect of changing the probability of dropout on performance. The architecture with dropping hidden unit activations with a probability of 0.1 during training gave the best performance, delivering 0.61–3.44% improvements over the other probability values tried.

4.3.4 Visual Inspection of RNN Activations

Figure 4.6 shows the activations in the RNN layer in the second stage for learning sequence-to-sequence interactions. The x-axis represents two RNA sequences (top: miRNA, bottom: mRNA), and the y-axis represents the position of each hidden unit in the layer. Using this plot, we can see how the activation changes over the nucleotide positions along the RNA sequences. Note that darker shades represents higher activation values. The red box indicates the activation sequence of the hidden unit that best represents the binding site pattern. This pattern seems compatible with the sequence align-

ment results shown by the two aligned RNA sequences in the x-axis. These results suggest that analyzing the activation patterns may allow us to gain novel biological insight into regulatory interactions.

4.4 Discussion

Biological sequences (such as DNA, RNA, and protein sequences) naturally fit the recurrent neural networks that are capable of temporal modeling. Nonetheless, prior work on applying deep learning to bioinformatics utilized only convolutional and fully connected neural networks. The biggest novelty of our work lies in its use of recurrent neural networks to model RNA sequences and further learn their sequence-to-sequence interactions, without laborious feature engineering (*e.g.*, more than 151 features of miRNA-target pairs have been proposed in the literature). As shown in our experimental results, even without any of the known features, deepTarget delivered substantial performance boosts (over 25% increase in F-measure) over existing miRNA target detectors, demonstrating the effectiveness of recent advances in end-to-end learning methodologies.

When dealing with genome-wide applications, such as miRNA target prediction, we often need to handle extremely imbalanced datasets in which negative examples significantly outnumber positive examples. According to Saito and Rehmsmeier [143], as PPV changes with the ratio of positives and negatives, PPV is more useful than the other metrics for evaluating binary classifiers on imbalanced datasets. As mentioned in Section 4.3.2, deepTarget was trained with 4,735 positive and 1,225 negative pairs and showed higher performance than the alternatives in terms of PPV. The higher values of sensitivity and PPV deepTarget reported from imbalance datasets indicate that deepTarget performs more robust prediction of miRNA-mRNA pairs.

When training deepTarget, we focused on improving its capability to reject false positives (*i.e.*, bogus miRNA-mRNA pairs) as a target predictor.

Given that there is a trade-off between sensitivity and specificity, our training principle resulted in slight loss in the specificity of deepTarget. This decision was based on the study that more priority should be given to sensitivity in the search for potential targets of specific miRNAs, whereas specificity should be emphasized in the examination of miRNAs that regulate specific genes [141]. Depending on the specific needs, we could alternatively train deepTarget to put more priority on specificity. For instance, this could be done by altering the composition of a mock negative dataset to have additional mispairings between miRNA and mRNA sequences except the seed sequence.

Notably, deepTarget does not depend on any sequence alignment operation, which has been used in many bioinformatics pipelines as a holy grail to reveal similarity/interactions between sequences. Although effective in general, sequence alignment is susceptible to changes in parameters (*e.g.*, gap/mismatch penalty and match premium) and often fails to reveal the true interactions between sequences, as is often observed in most of the alignment-based miRNA target detectors. By processing miRNA and RNA sequences with RNN-based auto-encoders without alignment, deepTarget successfully discover the inherent sequence representations, which are effectively used in the next step of deepTarget for interaction learning.

In Figure 4.6, we visualized the activations in the RNN layers of deepTarget as an attempt to discriminate between the positive and negative samples. Contrasting and scrutinizing the patterns existing in each class will be helpful for providing insight that can eventually lead to discovery of novel biological features. Given that biomedical practitioners prefer 'white-box' approaches (*e.g.*, logistic regression) to 'black-box' approaches (*e.g.*, deep learning), such efforts will hopefully expedite the deployment and acceptance of deep learning for biomedicine [144]. In this regard, attempts to decode the information learned by deep learning are emerging (*e.g.*, we can compute importance scores for each activation to better interpret activation energies [145]).

4.5 Summary

This chapter has presented an end-to-end machine learning framework for miRNA target prediction. MiRNAs are short sequences of ribonucleic acids that control the expression of target mRNAs by binding them. Robust prediction of miRNA-mRNA pairs is of utmost importance in deciphering gene regulation but has been challenging because of high false positive rates, despite a deluge of computational tools that normally require laborious manual feature extraction. Leveraged by deep recurrent neural networks-based auto-encoding and sequence-sequence interaction learning, our approach not only delivers an unprecedented level of accuracy but also eliminates the need for manual feature extraction. The performance gap between the proposed method and existing alternatives is substantial (over 25% increase in F-measure), and deepTarget delivers a quantum leap in the long-standing challenge of robust miRNA target prediction.

Chapter 5

Conclusion

To address the problems about errors induced during the sequencing procedures and unrevealed inherent features of biological data for inferring their interactions, this dissertation proposed a set of methodologies on the basis of machine learning algorithms for biological sequences that can boost the reliability of downstream analysis. We have discussed three issues in sequence analysis and proposed methodologies to overcome them.

Chapter 2 presented an information theoretic approach for correcting sequence errors from various sequencers. Our methodology is derived from a general setting of reconstructing finite-valued source data corrupted by a discrete memoryless channel and effectively corrects substitution and homopolymer indel errors. We approximate the empirical count vector of the underlying clean symbol (*i.e.*, clean nucleotide) at the middle location that resulted in the double-sided noisy context. We demonstrated the robustness and flexibility by showing that a simple change in channel matrix is enough to apply the exact same approach to data obtained using different sequencing platforms. In particular, we experimentally showed that even when the memoryless channel assumption does not hold, as in Illumina data, our methodology still solidly outperforms state-of-the-art schemes.

In Chapter 3, we showed a generalized multi-GPU accelerated sequence de-

noiser. To overcome the computational hurdle of the existing error-correction tools, we analyzed and exploited the data-level parallelism that exists in the error-correction procedure and proposed a tool that exploits both multi-core CPUs and GPUs for co-processing. According to our experimental results, as GPU-accelerated clusters provide the tremendous computing power required for computationally intensive workloads, our approach reduced not only the time demand for denoising amplicons, but also the number of overestimated OTUs.

In Chapter 4, we described an end-to-end machine learning framework for robust miRNA target prediction. The biggest novelty of our work lies in its use of RNNs to model RNA sequences and further learning their sequence-to-sequence interactions, without laborious feature engineering. Our approach adopts deep RNN-based auto-encoders to discover the inherent representations of miRNA and mRNA sequences and utilizes stacked RNNs to learn the sequence-to-sequence interactions underlying miRNA-mRNA pairs. As shown in our experimental results, even without any of the known features, our approach delivered substantial performance boosts over existing miRNA target detectors, demonstrating the effectiveness of recent advances in end-to-end learning methodologies.

There are also several possible extensions stemmed from the research in this dissertation. With respect to Chapter 2, we plan to test DUDE-Seq on several other sequencing platforms, such as PacBio and Oxford Nanopore, which tend to result in longer and more noisy sequences, to further substantiate the robustness and effectiveness of our algorithm. Applying the recently developed deep neural networks-based Neural DUDE algorithm [146] to DNA sequence denoising beyond targeted amplicon sequencing could be another fruitful research direction. With respect to Chapter 4, although the performance of deepTarget is incomparably higher than that of the compared state-of-the-art tools, there still remains room for further improvements. An additional breakthrough may

be possible by enhancing the current step to learn sequence-to-sequence interactions. The current version of deepTarget relies on concatenating the RNA representations from two auto-encoders and learning interactions therein using a unidirectional two-layer RNN architecture. Although this architecture was effective to some extents, as shown in our experiments, adopting more sophisticated approaches may even further boost the capability of deepTarget to detect subtle interactions that currently go undetected. Our future work thus includes investigating how to improve the interaction learning part, which is crucial for achieving additional performance gains.

Bibliography

[1] M. L. Metzker, "Sequencing technologies—the next generation," *Nature Reviews Genetics*, vol. 11, no. 1, pp. 31–46, 2010.

[2] W. T. Astbury, "Molecular biology or ultrastructural biology?," 1961.

[3] W. Bateson, *Materials for the Study of Variation, Treated with Especial Regard to Discontinuity in the Origin of Species*. Macmillan, 1894.

[4] C. S. Riesenfeld, P. D. Schloss, and J. Handelsman, "Metagenomics: genomic analysis of microbial communities," *Annu. Rev. Genet.*, vol. 38, pp. 525–552, 2004.

[5] M. Pop and S. L. Salzberg, "Bioinformatics challenges of new sequencing technology," *Trends in Genetics*, vol. 24, no. 3, pp. 142–149, 2008.

[6] J. Shendure and H. Ji, "Next-generation dna sequencing," *Nature biotechnology*, vol. 26, no. 10, pp. 1135–1145, 2008.

[7] S. Goodwin, J. D. McPherson, and W. R. McCombie, "Coming of age: ten years of next-generation sequencing technologies," *Nature Reviews Genetics*, vol. 17, no. 6, pp. 333–351, 2016.

[8] M. J. Bamshad, S. B. Ng, A. W. Bigham, H. K. Tabor, M. J. Emond, D. A. Nickerson, and J. Shendure, "Exome sequencing as a tool for mendelian disease gene discovery," *Nature Reviews Genetics*, vol. 12, no. 11, pp. 745–755, 2011.

[9] S. S. Jamuar and E.-C. Tan, "Clinical application of next-generation sequencing for mendelian diseases," *Human genomics*, vol. 9, no. 1, p. 1, 2015.

[10] X. Yang, S. P. Chockalingam, and S. Aluru, "A survey of error-correction methods for next-generation sequencing," *Briefings in bioinformatics*, vol. 14, no. 1, pp. 56–66, 2013.

[11] L. Ilie, F. Fazayeli, and S. Ilie, "Hitec: accurate error correction in high-throughput sequencing data," *Bioinformatics*, vol. 27, no. 3, pp. 295–302, 2011.

[12] W.-C. Kao, A. H. Chan, and Y. S. Song, "Echo: a reference-free short-read error correction algorithm," *Genome research*, vol. 21, no. 7, pp. 1181–1192, 2011.

[13] D. R. Kelley, M. C. Schatz, S. L. Salzberg, *et al.*, "Quake: quality-aware detection and correction of sequencing errors," *Genome Biol*, vol. 11, no. 11, p. R116, 2010.

[14] W. Qu, S.-i. Hashimoto, and S. Morishita, "Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing," *Genome research*, vol. 19, no. 7, pp. 1309–1315, 2009.

[15] L. Salmela, "Correction of sequencing errors in a mixed set of reads," *Bioinformatics*, vol. 26, no. 10, pp. 1284–1290, 2010.

[16] L. Salmela and J. Schröder, "Correcting errors in short reads by multiple alignments," *Bioinformatics*, vol. 27, no. 11, pp. 1455–1461, 2011.

[17] J. Schröder, H. Schröder, S. J. Puglisi, R. Sinha, and B. Schmidt, "Shrec: a short-read error correction method," *Bioinformatics*, vol. 25, no. 17, pp. 2157–2163, 2009.

[18] E. Wijaya, M. C. Frith, Y. Suzuki, and P. Horton, "Recount: expectation maximization based error correction tool for next generation sequencing data," in *Genome Inform*, vol. 23, pp. 189–201, World Scientific, 2009.

[19] X. Yang, S. Aluru, and K. S. Dorman, "Repeat-aware modeling and correction of short read errors," *BMC bioinformatics*, vol. 12, no. Suppl 1, p. S52, 2011.

[20] X. Yang, K. S. Dorman, and S. Aluru, "Reptile: representative tiling for short read error correction," *Bioinformatics*, vol. 26, no. 20, pp. 2526–2533, 2010.

[21] D. Laehnemann, A. Borkhardt, and A. C. McHardy, "Denoising dna deep sequencing data-high-throughput sequencing errors and their correction," *Briefings in Bioinformatics*, vol. 17, no. 1, pp. 154–179, 2016.

[22] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdú, and M. J. Weinberger, "Universal discrete denoising: Known channel," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 5–28, 2005.

[23] Z. Zheng *et al.*, "CRiSPy-CUDA: Computing species richness in 16S rRNA pyrosequencing datasets with CUDA," *Pattern Recognition in Bioinformatics*, pp. 37–49, 2011.

[24] H. Shi *et al.*, "A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware," *Journal of Computational Biology*, vol. 17, no. 4, pp. 603–615, 2010.

[25] C. Quince *et al.*, "Accurate determination of microbial diversity from 454 pyrosequencing data.," *Nature methods*, vol. 6, pp. 639–41, Sept. 2009.

[26] J. Reeder and R. Knight, "Rapidly denoising pyrosequencing amplicon reads by exploiting rank-abundance distributions.," *Nature methods*, vol. 7, pp. 668–9, Sept. 2010.

[27] C. Quince *et al.*, "Removing noise from pyrosequenced amplicons.," *BMC bioinformatics*, vol. 12, p. 38, Jan. 2011.

[28] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," in *Computer graphics forum*, vol. 26, pp. 80–113, Wiley Online Library, 2007.

[29] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE micro*, vol. 30, no. 2, pp. 56–69, 2010.

[30] P. Jia, L. Xuan, L. Liu, C. Wei, and J. H. Badger, "Metabing: Using gpus to accelerate metagenomic sequence classification," *PloS one*, vol. 6, no. 11, p. e25353, 2011.

[31] P. D. Vouzis and N. V. Sahinidis, "Gpu-blast: using graphics processors to accelerate protein sequence alignment," *Bioinformatics*, vol. 27, no. 2, pp. 182–188, 2011.

[32] R. C. Friedman, K. K.-H. Farh, C. B. Burge, and D. P. Bartel, "Most mammalian mRNAs are conserved targets of microRNAs," *Genome Research*, vol. 19, no. 1, pp. 92–105, 2009.

[33] H. Min and S. Yoon, "Got target?: Computational methods for microRNA target prediction and their extension," *Experimental & Molecular Medicine*, vol. 42, no. 4, pp. 233–244, 2010.

[34] S. M. Peterson, J. A. Thompson, M. L. Ufkin, P. Sathyanarayana, L. Liaw, and C. B. Congdon, "Common features of microRNA target prediction tools," *Front Genet*, vol. 5, p. 23, 2014.

[35] S. Yoon and G. De Micheli, "Computational identification of microRNAs and their targets," *Birth Defects Research Part C: Embryo Today: Reviews*, vol. 78, no. 2, pp. 118–128, 2006.

[36] M. Menor, T. Ching, X. Zhu, D. Garmire, and L. X. Garmire, "mirMark: a site-level and UTR-level classifier for miRNA target prediction," *Genome biology*, vol. 15, no. 10, p. 500, 2014.

[37] P. Medvedev, E. Scott, B. Kakaradov, and P. Pevzner, "Error correction of high-throughput sequencing datasets with non-uniform coverage," *Bioinformatics*, vol. 27, no. 13, pp. i137–i141, 2011.

[38] S. I. Nikolenko, A. I. Korobeynikov, and M. A. Alekseyev, "Bayeshammer: Bayesian clustering for error correction in single-cell sequencing," *BMC genomics*, vol. 14, no. Suppl 1, p. S7, 2013.

[39] P. Greenfield, K. Duesing, A. Papanicolaou, and D. C. Bauer, "Blue: correcting sequencing errors using consensus and context," *Bioinformatics*, vol. 30, no. 19, pp. 2723–2732, 2014.

[40] E.-C. Lim, J. Müller, J. Hagmann, S. R. Henz, S.-T. Kim, and D. Weigel, "Trowel: a fast and accurate error correction module for illumina sequencing reads," *Bioinformatics*, p. btu513, 2014.

[41] L. Bragg, G. Stone, M. Imelfort, P. Hugenholtz, and G. W. Tyson, "Fast, accurate error-correction of amplicon pyrosequences using acacia," *Nature Methods*, vol. 9, no. 5, pp. 425–426, 2012.

[42] F. Meacham, D. Boffelli, J. Dhahbi, D. I. Martin, M. Singer, and L. Pachter, "Identification and correction of systematic error in high-throughput sequence data," *BMC bioinformatics*, vol. 12, no. 1, p. 451, 2011.

[43] X. Yin, Z. Song, K. Dorman, and A. Ramamoorthy, "Premier—probabilistic error-correction using markov inference in errored reads," in *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1626–1630, 2013.

[44] M. H. Schulz, D. Weese, M. Holtgrewe, V. Dimitrova, S. Niu, K. Reinert, and H. Richard, "Fiona: a parallel and automatic strategy for read error correction," *Bioinformatics*, vol. 30, no. 17, pp. i356–i363, 2014.

[45] B. Ewing, L. Hillier, M. C. Wendl, and P. Green, "Base-calling of automated sequencer traces using phred. i. accuracy assessment," *Genome research*, vol. 8, no. 3, pp. 175–185, 1998.

[46] M. Schirmer, U. Z. Ijaz, R. D'Amore, N. Hall, W. T. Sloan, and C. Quince, "Insight into biases and sequencing errors for amplicon sequencing with the illumina miseq platform," *Nucleic acids research*, p. gku1341, 2015.

[47] B. Yan, Y. Hu, C. Ng, K. H. Ban, T. W. Tan, P. T. Huan, P.-L. Lee, L. Chiu, E. Seah, C. H. Ng, *et al.*, "Coverage analysis in a targeted amplicon-based next-generation sequencing panel for myeloid neoplasms," *Journal of clinical pathology*, pp. jclinpath–2015, 2016.

[48] A. Dembo and T. Weissman, "Universal denoising for the finite-input general-output channel," *Information Theory, IEEE Transactions on*, vol. 51, no. 4, pp. 1507–1517, 2005.

[49] C. Quince, A. Lanzen, R. J. Davenport, and P. J. Turnbaugh, "Removing noise from pyrosequenced amplicons," *BMC bioinformatics*, vol. 12, no. 1, p. 38, 2011.

[50] L. M. Bragg, G. Stone, M. K. Butler, P. Hugenholtz, and G. W. Tyson, "Shining a light on dark sequencing: characterising errors in ion torrent pgm data," *PLoS Comput Biol*, vol. 9, no. 4, p. e1003031, 2013.

[51] E. B. Fichot and R. S. Norman, "Microbial phylogenetic profiling with the pacific biosciences sequencing platform," *Microbiome*, vol. 1, no. 1, p. 10, 2013.

[52] E. Marinier, D. G. Brown, and B. J. McConkey, "Pollux: platform independent error correction of single and mixed genomes," *BMC bioinformatics*, vol. 16, no. 1, p. 10, 2015.

[53] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[54] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, *et al.*, "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[55] M. C. LaFave and S. M. Burgess, "sam2pairwise version 1.0.0," Aug. 2014.

[56] D. Pfitzner, R. Leibbrandt, and D. Powers, "Characterization and evaluation of similarity measures for pairs of clusterings," *Knowledge and Information Systems*, vol. 19, no. 3, pp. 361–394, 2009.

[57] J. Reeder and R. Knight, "Rapid denoising of pyrosequencing amplicon data: exploiting the rank-abundance distribution," *Nature methods*, vol. 7, no. 9, p. 668, 2010.

[58] P. D. Schloss and J. Handelsman, "Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness," *Applied and environmental microbiology*, vol. 71, no. 3, pp. 1501–1506, 2005.

[59] R. C. Edgar, "Search and clustering orders of magnitude faster than blast," *Bioinformatics*, vol. 26, no. 19, pp. 2460–2461, 2010.

[60] A. K. Bartram, M. D. Lynch, J. C. Stearns, G. Moreno-Hagelsieb, and J. D. Neufeld, "Generation of multimillion-sequence 16s rrna gene libraries from complex microbial communities by assembling paired-end

illumina reads," *Applied and environmental microbiology*, vol. 77, no. 11, pp. 3846–3852, 2011.

[61] T. Magoč and S. L. Salzberg, "Flash: fast length adjustment of short reads to improve genome assemblies," *Bioinformatics*, vol. 27, no. 21, pp. 2957–2963, 2011.

[62] Y. Heo, X.-L. Wu, D. Chen, J. Ma, and W.-M. Hwu, "Bless: bloom filter-based error correction solution for high-throughput sequencing reads," *Bioinformatics*, p. btu030, 2014.

[63] H. Li, "Exploring single-sample snp and indel calling with whole-genome de novo assembly," *Bioinformatics*, vol. 28, no. 14, pp. 1838–1844, 2012.

[64] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965.

[65] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[66] F. E. Angly, D. Willner, F. Rohwer, P. Hugenholtz, and G. W. Tyson, "Grinder: a versatile amplicon and shotgun sequence simulator," *Nucleic acids research*, vol. 40, no. 12, pp. e94–e94, 2012.

[67] K. E. McElroy, F. Luciani, and T. Thomas, "Gemsim: general, error-model based simulator of next-generation sequencing data," *BMC genomics*, vol. 13, no. 1, p. 74, 2012.

[68] S. Kwon, B. Lee, and S. Yoon, "Casper: context-aware scheme for paired-end reads from high-throughput amplicon sequencing," *BMC bioinformatics*, vol. 15, no. Suppl 9, p. S10, 2014.

[69] B. Liu, J. Yuan, S.-M. Yiu, Z. Li, Y. Xie, Y. Chen, Y. Shi, H. Zhang, Y. Li, T.-W. Lam, *et al.*, "Cope: an accurate k-mer-based pair-end reads

connection tool to facilitate genome assembly," *Bioinformatics*, vol. 28, no. 22, pp. 2870–2874, 2012.

[70] A. P. Masella, A. K. Bartram, J. M. Truszkowski, D. G. Brown, and J. D. Neufeld, "Pandaseq: paired-end assembler for illumina sequences," *BMC bioinformatics*, vol. 13, no. 1, p. 31, 2012.

[71] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[72] M. Jain, I. T. Fiddes, K. H. Miga, H. E. Olsen, B. Paten, and M. Akeson, "Improved data analysis for the minion nanopore sequencer," *Nature methods*, 2015.

[73] G. M. Gemelos, S. Sigurjonsson, and T. Weissman, "Algorithms for discrete denoising under channel uncertainty," *Signal Processing, IEEE Transactions on*, vol. 54, no. 6, pp. 2263–2276, 2006.

[74] I. Ochoa, M. Hernaez, R. Goldfeder, E. Ashley, and T. Weissman, "Effect of lossy compression of quality scores on variant calling," *Bioinformatics, under review*, 2016.

[75] J. Qin *et al.*, "A human gut microbial gene catalogue established by metagenomic sequencing.," *Nature*, vol. 464, pp. 59–65, Mar. 2010.

[76] P. Eckburg *et al.*, "Diversity of the human intestinal microbial flora," *Science*, vol. 308, no. 5728, pp. 1635–1638, 2005.

[77] P. Lorenz and J. Eck, "Metagenomics and industrial applications," *Nature Reviews Microbiology*, vol. 3, no. 6, pp. 510–516, 2005.

[78] J. G. Caporaso *et al.*, "QIIME allows analysis of high- throughput community sequencing data Intensity normalization improves color calling in SOLiD sequencing," *Nature methods*, vol. 7, no. 5, pp. 335–336, 2010.

[79] M. Pop and S. L. Salzberg, "Bioinformatics challenges of new sequencing technology," *Trends in Genetics*, vol. 24, no. 3, pp. 142–149, 2008.

[80] V. Kunin *et al.*, "Wrinkles in the rare biosphere: pyrosequencing errors can lead to artificial inflation of diversity estimates.," *Environmental microbiology*, vol. 12, pp. 118–23, Jan. 2010.

[81] Y. Liu, B. Schmidt, and D. L. Maskell, "Decgpu: distributed error correction on massively parallel graphics processing units using cuda and mpi," *BMC bioinformatics*, vol. 12, no. 1, p. 85, 2011.

[82] M. Mysara, N. Leys, J. Raes, and P. Monsieurs, "Node: a fast error-correction algorithm for pyrosequencing amplicon reads," *BMC bioinformatics*, vol. 16, no. 1, p. 88, 2015.

[83] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, pp. 1–38, 1977.

[84] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[85] R. Farber, *CUDA application design and development*. Morgan Kaufmann Pub, 2011.

[86] S. A. Manavski and G. Valle, "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment.," *BMC bioinformatics*, vol. 9 Suppl 2, p. S10, Jan. 2008.

[87] J. Blazewicz *et al.*, "Protein alignment algorithms with an efficient backtracking routine on multiple GPUs," *BMC bioinformatics*, vol. 12, p. 181, 2011.

[88] M. C. Schatz *et al.*, "High-throughput sequence alignment using Graphics Processing Units," *BMC Bioinformatics*, vol. 8, p. 474, 2007.

[89] B. Lee, J. Park, and S. Yoon, "Rapid and robust denoising of pyrosequenced amplicons for metagenomics," in *Proceedings of the IEEE 12th International Conference on Data Mining (ICDM)*, pp. 954–959, 2012.

[90] C. Fraley and A. Raftery, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998.

[91] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Boston, MA: Pearson Addison Wesley, 2006.

[92] J. Eichorn, *Understanding AJAX: Using JavaScript to create rich internet applications*. Prentice Hall PTR, 2006.

[93] F. Sievers *et al.*, "Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega," *Molecular systems biology*, vol. 7, no. 1, 2011.

[94] N. P. Brown, C. Leroy, and C. Sander, "Mview: a web-compatible database search or multiple alignment viewer.," *Bioinformatics*, vol. 14, no. 4, pp. 380–381, 1998.

[95] N. Kumar, S. Satoor, and I. Buck, "Fast parallel expectation maximization for gaussian mixture models on gpus using cuda," in *High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on*, pp. 103–109, IEEE, 2009.

[96] G. Shainer, A. Ayoub, P. Lui, T. Liu, M. Kagan, C. R. Trott, G. Scantlen, and P. S. Crozier, "The development of mellanox/nvidia gpudirect over infiniband—a new model for gpu to gpu communications," *Computer Science-Research and Development*, vol. 26, no. 3-4, pp. 267–273, 2011.

[97] P. Rogers, J. Macri, and S. Marinkovic, "Amd heterogeneous uniform memory access," 2013.

[98] J. F. Petrosino, S. Highlander, R. A. Luna, R. A. Gibbs, and J. Versalovic, "Metagenomic pyrosequencing and microbial identification," *Clinical chemistry*, vol. 55, no. 5, pp. 856–866, 2009.

[99] C. Luo, D. Tsementzi, N. Kyrpides, T. Read, and K. T. Konstantinidis, "Direct comparisons of illumina vs. roche 454 sequencing technologies on the same microbial community dna sample," *PloS one*, vol. 7, no. 2, p. e30087, 2012.

[100] R. Logares, S. Sunagawa, G. Salazar, F. M. Cornejo-Castillo, I. Ferrera, H. Sarmento, P. Hingamp, H. Ogata, C. Vargas, G. Lima-Mendez, *et al.*, "Metagenomic 16s rdna illumina tags are a powerful alternative to amplicon sequencing to explore diversity and structure of microbial communities," *Environmental microbiology*, vol. 16, no. 9, pp. 2659–2671, 2014.

[101] D. P. Bartel, "MicroRNAs: genomics, biogenesis, mechanism, and function," *Cell*, vol. 116, no. 2, pp. 281–297, 2004.

[102] J. Lu, G. Getz, E. A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B. L. Ebert, R. H. Mak, A. A. Ferrando, *et al.*, "MicroRNA expression profiles classify human cancers," *nature*, vol. 435, no. 7043, pp. 834–838, 2005.

[103] P. K. Srivastava, T. R. Moturu, P. Pandey, I. T. Baldwin, and S. P. Pandey, "A comparison of performance of plant miRNA target prediction tools and the characterization of features for genome-wide target prediction," *BMC Genomics*, vol. 15, no. 1, p. 1, 2014.

[104] S. Cheng, M. Guo, C. Wang, X. Liu, Y. Liu, and X. Wu, "MiRTDL: a deep learning approach for miRNA target prediction,"

[105] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, ACM, 2008.

[106] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[107] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "DRAW: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[108] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[109] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[110] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[111] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[112] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[113] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[114] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[115] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[116] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[117] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *arXiv preprint arXiv:1411.4555*, 2014.

[118] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[119] P. Baldi and S. Brunak, "Chapter 6. neural networks: applications," in *Bioinformatics: The Machine Learning Approach*, MIT press, 2001.

[120] B. Lee, T. Lee, B. Na, and S. Yoon, "DNA-level splice junction prediction using deep recurrent neural networks," *arXiv preprint arXiv:1512.05135*, 2015.

[121] T. Lee and S. Yoon, "Boosted categorical restricted boltzmann machine for computational prediction of splice junctions," in *ICML*, pp. 2483—-2492, 2015.

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

[123] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," *arXiv preprint arXiv:1502.04681*, 2015.

[124] P. Maziere and A. J. Enright, "Prediction of microRNA targets," *Drug discovery today*, vol. 12, no. 11, pp. 452–458, 2007.

[125] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[126] K. Sohn, W. Shang, and H. Lee, "Improved multimodal deep learning with variation of information," in *Advances in Neural Information Processing Systems*, pp. 2141–2149, 2014.

[127] F. Xiao, Z. Zuo, G. Cai, S. Kang, X. Gao, and T. Li, "miRecords: an integrated resource for microRNA–target interactions," *Nucleic Acids Research*, vol. 37, no. suppl 1, pp. D105–D110, 2009.

[128] S. Griffiths-Jones, H. K. Saini, S. van Dongen, and A. J. Enright, "miRBase: tools for microRNA genomics," *Nucleic Acids Research*, vol. 36, no. suppl 1, pp. D154–D158, 2008.

[129] B. John, A. J. Enright, A. Aravin, T. Tuschl, C. Sander, D. S. Marks, *et al.*, "Human microRNA targets," *PLoS Biol*, vol. 2, no. 11, p. e363, 2004.

[130] M. Maragkakis, P. Alexiou, G. L. Papadopoulos, M. Reczko, T. Dalamagas, G. Giannopoulos, G. Goumas, E. Koukis, K. Kourtis, V. A. Simossis, *et al.*, "Accurate microRNA target prediction correlates with protein repression levels," *BMC Bioinformatics*, vol. 10, no. 1, p. 295, 2009.

[131] R. A. Fisher, F. Yates, *et al.*, "Statistical tables for biological, agricultural and medical research.," *Statistical tables for biological, agricultural and medical research.*, no. Ed. 3., 1949.

[132] A. J. Enright, B. John, U. Gaul, T. Tuschl, C. Sander, D. S. Marks, *et al.*, "MicroRNA targets in Drosophila," *Genome Biology*, vol. 5, no. 1, pp. R1–R1, 2004.

[133] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[134] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[135] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[136] S. Bandyopadhyay, D. Ghosh, R. Mitra, and Z. Zhao, "MBSTAR: multiple instance learning for predicting specific functional binding sites in microRNA targets," *Scientific reports*, vol. 5, 2015.

[137] M. Kertesz, N. Iovino, U. Unnerstall, U. Gaul, and E. Segal, "The role of site accessibility in microRNA target recognition," *Nature Genetics*, vol. 39, no. 10, pp. 1278–1284, 2007.

[138] K. C. Miranda, T. Huynh, Y. Tay, Y.-S. Ang, W.-L. Tam, A. M. Thomson, B. Lim, and I. Rigoutsos, "A pattern-based method for the identification of microRNA binding sites and their corresponding heteroduplexes," *Cell*, vol. 126, no. 6, pp. 1203–1217, 2006.

[139] B. P. Lewis, I.-h. Shih, M. W. Jones-Rhoades, D. P. Bartel, and C. B. Burge, "Prediction of mammalian microRNA targets," *Cell*, vol. 115, no. 7, pp. 787–798, 2003.

[140] M. Sturm, M. Hackenberg, D. Langenberger, and D. Frishman, "TargetSpy: a supervised machine learning approach for microRNA target prediction," *BMC Bioinformatics*, vol. 11, no. 1, p. 292, 2010.

[141] T. M Witkos, E. Koscianska, and W. J Krzyzosiak, "Practical aspects of microRNA target prediction," *Current molecular medicine*, vol. 11, no. 2, pp. 93–109, 2011.

[142] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2342–2350, 2015.

[143] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PloS One*, vol. 10, no. 3, p. e0118432, 2015.

[144] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in Bioinformatics*, in press, 2016.

[145] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.

[146] T. Moon, S. Min, B. Lee, and S. Yoon, "Neural universal discrete denoiser," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2016.

초 록

인간 게놈 프로젝트가 완료된 이래로 인간의 생물학적 매커니즘을 이해하려는 시도 중의 하나로서 방대한 양의 생물학적 데이터가 축적되었다. 하지만 염기서열 분석 과정에서 혼입된 오류와 생물학적 데이터 간의 상호 관계를 추론하는 데 필요한 밝혀지지 않은 내재된 특징들로 인하여 대규모 데이터 기반 도구의 필요성이 증대되고 있다. 이러한 점에서 본 논문은 최근 자연어 처리와 신경망 번역 등 시계열 데이터 학습에 우수한 성능을 보이고 있는 기계학습과 인공지능 기술의 진보를 생물학적 데이터 분석의 신뢰성과 컴퓨터를 이용한 분석 성능을 향상시키는 데 활용하였다.

본 논문은 서열 분석과 관련된 세 가지 사안을 논하고, 이를 극복하기 위한 방법론을 제안한다. 첫 번째로 차세대 염기서열 분석 과정에 내재된 오류를 완화하기 위하여 다양한 염기서열 분석 플랫폼에 적용 가능한 정보이론 기반의 오류 정정 접근법을 제안한다. 다음으로 고효율 서열 데이터의 연산 문제를 다루기 위한 다수의 그래픽 처리장치 기반의 일반화된 오류 정정 알고리즘을 제안한다. 마지막으로 수동적인 특징 추출 과정이 없이 민감도를 높인 서열 (예, miRNA) 결합 예측을 위한 종단 간 기계학습 프레임워크를 제안한다. 요약하면 본 논문은 생물학적 서열에 대한 후속 분석의 신뢰성을 높일 수 있는 기계학습 알고리즘에 기반한 일련의 방법론을 제안한다.

주요어: machine learning, deep learning, end-to-end learning, parallelization, sequence error, sequence interaction, time series, miRNA target

학번: 2012-23229