공학박사 학위논문

Representation Learning for Biological Sequence Data

생물학적 서열 데이터에 대한 표현 학습

2021년 8월

서울대학교 대학원

전기·정보공학부

민 선 우

Representation Learning for Biological Sequence Data

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함

2021년 8월

서울대학교 대학원

전기·정보공학부

민 선 우

민선우의 공학박사 학위논문을 인준함

2021년 8월

위 원 장 _____

부위원장 _____

위 원 _____

위 원 _____

위 원 _____

Abstract

As we are living in the era of big data, the biomedical domain is not an exception. With the advent of technologies such as next-generation sequencing, developing methods to capitalize on the explosion of biomedical data is one of the most major challenges in bioinformatics. Representation learning, in particular deep learning, has made significant advancements in diverse fields where the artificial intelligence community has struggled for many years. However, although representation learning has also shown great promises in bioinformatics, it is not a silver bullet. Off-the-shelf applications of representation learning cannot always provide successful results for biological sequence data. There remain full of challenges and opportunities to be explored.

This dissertation presents a set of representation learning methods to address three issues in biological sequence data analysis. First, we propose a two-stage training strategy to address throughput and information trade-offs within wet-lab CRISPR-Cpf1 activity experiments. Second, we propose an encoding scheme to model interaction between two sequences for functional microRNA target prediction. Third, we propose a self-supervised pre-training method to bridge the exponentially growing gap between the numbers of unlabeled and labeled protein sequences. In summary, this dissertation proposes a set of representation learning methods that can derive invaluable information from the biological sequence data.

Keywords: machine learning, deep learning, representation learning, artificial intelligence, biological sequence, CRISPR, microRNA target, protein

Student Number: 2015-20924

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As we are living in the era of big data, the biomedical domain is not an exception. With the advent of advanced technologies such as next-generation sequencing, significant amounts of biomedical data have been accumulated. The biomedical data generation has been even exceeding researchers' ability to capitalize on the data [1]. The transformation of big data into valuable knowledge is one of the significant challenges in bioinformatics.

Machine learning has been one of the most widely used and successful methodologies to extract valuable knowledge from data. In contrast to other methods, machine learning algorithms do not require strong assumptions on underlying biological mechanisms [2]. Instead, they use large quantities of data to uncover underlying patterns, build predictive models, and make predictions with the fitted model for unseen data. Various conventional machine learning algorithms (*e.g.*, random forests, support vector machines (SVMs), Bayesian networks, and hidden Markov models (HMMs)) have been widely applied in diverse problems in bioinformatics [3]. However, their performance relies heavily on data representations called features. It typically requires human engineers with the extensive domain expertise to design the hand-crafted features, and

identifying effective features for a variety of tasks remains labor-intensive and time-consuming processes [4].

Representation learning is one of the approaches to solve this central problem in conventional machine learning algorithms. Representation learning algorithms eliminate the laborious feature engineering and learn effective representations solely from data [5]. In particular, deep learning is a branch of representation learning, which uses deep neural networks (DNNs) with multiple nonlinear operations. Deep learning algorithms can learn complex hierarchical data representations built out of simpler concepts with an increasing level of abstraction. Based on big data, efficient computing, and sophisticated training algorithms, deep learning algorithms have made significant advancements in diverse fields (*e.g.*, image recognition, speech recognition, and natural language processing) where the artificial intelligence community has struggled for many years [6].

As summarized in our recent review paper [7], there also have been exponentially growing interests in deep learning for bioinformatics. For solving problems in a variety of bioinformatics domains (*e.g.*, omics, biomedical imaging, and biomedical signal processing), researchers have used different deep learning model architectures *e.g.*, multi-layer perceptrons (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers (TFMs) based on data characteristics and research objectives. Although deep learning has shown great promise, it is not a silver bullet. Off-the-shelf applications of deep learning algorithms cannot always provide successful results, especially for biological sequence data analyses. There remain full of challenges and opportunities to be explored [8].

In this dissertation, we present a set of representation learning methods to analyze biological sequence data, *i.e.*, deoxyribonucleic acids (DNAs), ribonucleic acids (RNAs), and proteins. A sequence can be defined as a series of items where their orderings as well as identities of each item contain crit-

ical information. For example, DNAs and RNAs are composed of a series of nucleotides; proteins are composed of a series of amino acids. Various representation learning methods have been used to derive invaluable information from the biological sequence data [7]. In this dissertation, we focus on addressing three issues in biological sequence data analysis: (1) throughput and information trade-offs within wet-lab experiments, (2) modeling interaction between two sequences, and (3) exponentially growing gap between the numbers of unlabeled and labeled protein sequences.

1.2 Contents of Dissertation

The contents of the dissertation are organized as follows. In Chapter 2, we cover the background knowledge necessary to follow the main contributions. In Chapters 3 - 5, we first introduce a bioinformatics problem and core issues in previous machine learning approaches. Then, we propose novel representation learning methods, which can address the core issues and derive invaluable information from the biological sequence data. In Chapter 6, we discuss some limitations of the proposed methods and promising research extensions for future work. Finally, Chapter 7 concludes the dissertation and summarizes the contributions towards analyzing biological sequence data with representation learning.

Chapter 3 addresses throughput and information trade-offs within wet-lab experiments. Over the last few years, genome editing using CRISPR (clustered, regularly interspaced, short palindromic repeats) system has become a crucial tool in biology [9]. One of the biggest challenges of CRISPR technology is the determination of nuclease activity [10]. There are two critical genetic and epigenetic factors that are known to affect the CRISPR activity [11]. However, due to the low-throughput wet-lab experiments, it is difficult to obtain endogenous target datasets which reflect both factors to train a model. Instead, we can obtain high-throughput integrated target datasets which only reflect the genetic factor. There are several shallow learning-based algorithms that enable the prediction of CRISPR activity. Nevertheless, they rely on manual feature extraction and do not consider the epigenetic factor, which inevitably reduces their reliability [12, 11].

This dissertation proposes an end-to-end deep learning framework and two-stage training strategy to address the throughput and information trade-offs within wet-lab CRISPR-Cpf1 activity experiments. First, we used integrated target datasets to train a deep-learning framework, dubbed as Seq-deepCpf1. Then, we used endogenous target datasets to incorporate chromatin accessibil-

ity information into a better-performing model, dubbed as DeepCpf1. We take advantage of (1) a CNN for feature learning from target sequence composition and (2) a multi-modal architecture for seamless integration of both genetic and epigenetic factors. The proposed methods significantly outperform the conventional machine learning-based approaches with an unprecedented level of high accuracy. The contents of this Chapter are based on the following research:

- Hui Kwon Kim*, Seonwoo Min*, Myungjae Song, Soobin Jung, Jae Woo Choi, Younggwang Kim, Sangeun Lee, Sungroh Yoon, and Hyongbum Kim, "Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity," *Nature Biotechnology*, vol. 36, no. 3, pp. 239-241, March 2018. (*co-first authors)

Chapter 4 addresses modeling interaction between two biological sequences. MicroRNAs play pivotal roles in gene expression regulation by binding to target sites of messenger RNAs (mRNAs) [13]. While identifying functional targets of microRNAs is of utmost importance, previous computational algorithms share some major limitations [14, 15]. First, they use conservative candidate target site (CTS) selection criteria mainly focusing on canonical site types. Second, they rely on laborious and time-consuming manual feature extraction. Third, they do not fully capitalize on the information underlying microRNA-CTS interactions.

This dissertation proposes a deep learning-based algorithm with a novel encoding scheme to model the interaction between two biological sequences. To address the limitations of previous approaches, the proposed TargetNet has three key components: (1) relaxed CTS selection criteria accommodating more irregularities in the seed region, (2) a novel microRNA-CTS sequence encoding scheme incorporating extended seed region alignment results, and (3) a deep residual CNN-based prediction model. The proposed model was trained with microRNA-CTS pair datasets and evaluated with microRNA-mRNA pair datasets. Our experiment results showed that TargetNet advances the previ-

ous state-of-the-art (SOTA) algorithms used in functional microRNA target classification. Furthermore, it demonstrates great potential for distinguishing high-functional microRNA targets. The contents of this Chapter are based on the following research:

- Seonwoo Min, Byunghan Lee, and Sungroh Yoon, "TargetNet: Functional microRNA target prediction with deep neural networks," *Bioinformatics*, under review.

Chapter 5 addresses the exponentially growing gap between the numbers of unlabeled and labeled protein sequences. With the development of next-generation sequencing technologies, protein sequences have become relatively more accessible. However, annotating a sequence with meaningful attributes such as its structures and functions is still time-consuming and resource-intensive [16]. Several studies have adopted semi-supervised learning for protein sequence modeling [17, 18]. They pre-train models with a substantial amount of unlabeled protein sequences, and the representations are transferred to various downstream tasks with a small number of labeled protein sequences. They have shown that pre-training helps improving prediction performance in downstream protein biology tasks. However, most pre-training methods solely rely on language modeling and still often exhibit limited performance [19].

This dissertation proposes a pre-training method to bridge the exponentially growing gap between the numbers of unlabeled and labeled protein sequences. We introduce a novel pre-training method for protein sequence modeling and name it PLUS, which stands for Protein sequence representations Learned Using Structural information. The proposed PLUS consists of masked language modeling and a complementary protein-specific pre-training task, namely same-family prediction. PLUS can be used to pre-train various model architectures. In this work, we use it to pre-train a bidirectional RNN (BiRNN) and refer to the resulting model as PLUS-RNN. Our experiment results demonstrate that PLUS-RNN outperforms other models of similar size

solely pre-trained with the language modeling in six out of seven widely used protein biology tasks. It provides a novel way to exploit evolutionary relationships among unlabeled proteins and is broadly applicable across various protein biology tasks. The contents of this Chapter are based on the following research:

- Seonwoo Min, Seunghyun Park, Siwon Kim, Hyun-Soo Choi, and Sungroh Yoon, "Pre-training of deep bidirectional protein sequence representations with structural information," in *the 14th Annual Workshop on Machine Learning in Computational Biology*, Vancouver, Canada, December 2019.

- Seonwoo Min, Hyungi Kim, Byunghan Lee, and Sungroh Yoon, "Protein transfer learning improves identification of heat shock protein families," *PLOS ONE*, vol. 16, no. 5, e0251865, May 2021.

Chapter 2

Background

In this Chapter, we cover the background knowledge necessary to follow the main contributions of the dissertation. We will explain an overview of representation learning, various DNN architectures (*i.e.*, MLPs, CNNs, RNNs, and TFMs), training of DNNs, previous works using representation learning methods in bioinformatics, fundamentals of biological sequence data analyses, and common evaluation metrics.

2.1 Representation Learning

This section reviews different disciplines, including artificial intelligence, machine learning, representation learning, and deep learning. We note that the contents of this section are referenced from the *Deep Learning* textbook [5]. Efforts to create artificial intelligence algorithms have a long history. Figure 2.1 shows relationships between different artificial intelligence disciplines. In the early days of artificial intelligence, people have succeeded in solving various problems. Although these problems were intellectually difficult for humans, they did not require any complex and general understanding of the world. Most of the problems could be easily defined with formal rules, making it relatively straightforward for computer algorithms. For example, one of the most

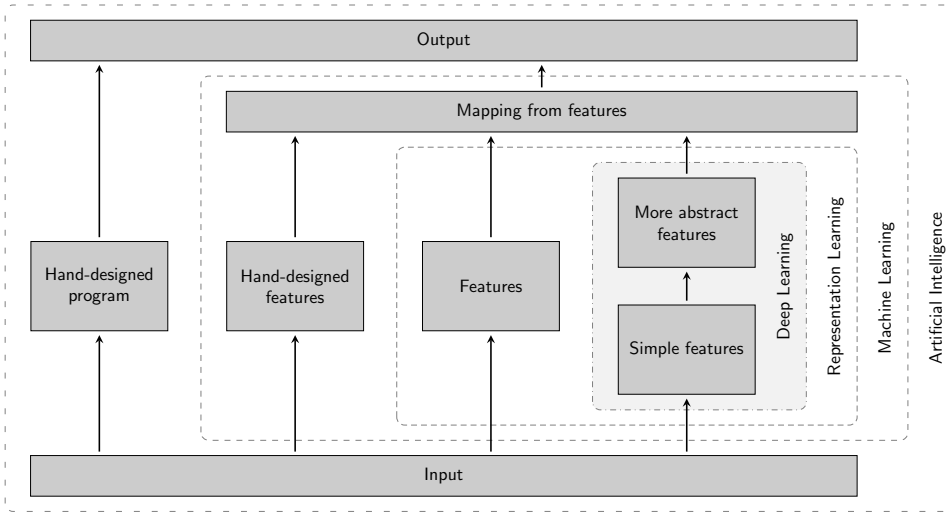

Figure 2.1: A diagram showing relationships of different artificial intelligence disciplines, *i.e.*, artificial intelligence, machine learning, representation learning, and deep learning

popular early successes is IBM's Deep Blue for playing chess [20]. It was a huge milestone in the history of artificial intelligence. However, the challenge at that time was very different from the one we are facing today. Chess can be easily described with pre-defined rules containing how each piece can move rigidly.

The challenge we are facing today with artificial intelligence comes from problems that we cannot formally describe the rules. Humans can solve them intuitively without acknowledging detailed logic. However, these problems require a tremendous amount of knowledge about the world, which is subjective and difficult to articulate definitely. For example, think about the image recognition problem, particularly classifying images of cats and dogs. Humans, even a child, can easily classify images, but it is not easy to formally define which images should be classified as cats or dogs. One of the critical challenges is how to deliver such informal knowledge to artificial intelligence algorithms. Early approaches tried to explicitly devise hard-coded rules for given tasks. Nonetheless, hand-designing all the rules to deal with complex real-world problems was

nearly impossible and could not achieve satisfactory results.

Machine learning has provided a viable solution to the challenge of artificial intelligence. Instead of hard-coded knowledge, we can provide data representations known as features. Then, given a training dataset, machine learning algorithms learn mappings from the features to target outcomes. For example, in the image recognition problem, we may provide the presence or absence of pointy ears as a feature. This feature alone is not enough to classify the images of cats and dogs, but combining with other numerous features would be able to provide the required information. The critical aspect of machine learning algorithms is extracting the right set of effective features. The algorithms cannot influence the way features are extracted, so they cannot make any useful predictions without effective features. Unfortunately, however, it is also not easy to hand-design effective features for given tasks. We may know the presence or absence of pointy ears would be a good feature, but it is still difficult to precisely define it in terms of pixel values. Furthermore, for more complex tasks in biological sequence data analyses, we often do not have enough domain knowledge to hand-design the effective features.

Naturally, the next step for more advanced artificial intelligence was to learn representation themselves from data, as well as their mappings to target outputs. This type of algorithm is known as representation learning. One most widely used example is an autoencoder. It consists of an encoder that converts data into representations and a decoder that converts the representations back into the data. By training the autoencoder to reconstruct data as much as possible, the learned representations would contain a wealth of information of data. The learned representations can often provide better performance than hand-designed ones, which require a great deal of time, effort, and domain expertise.

The challenge of a simple autoencoder is that it is difficult to learn complex high-level representations from raw data. Recently, deep learning, a branch of

representation learning, has tackled this challenge by introducing DNNs. Deep learning algorithms enable learning hierarchical representations of data with increasing levels of abstraction, where complex representations are built out of simpler representations. The DNNs are composed of multiple nonlinear operations gradually transform raw pixels into an informative representation. For example, for the image recognition problem, we can understand deep learning algorithms as learning to find edges from raw pixels, then contours from the edges, object parts from the contours, and finally recognize object identities from the object parts.

In summary, machine learning is currently the most viable approach to build artificial intelligence systems [5]. It allows us to overcome the limitations of hard-coded knowledge and improve with data. In this dissertation, we focus on representation learning, particularly deep learning algorithms, which hold excellent capability by learning to encode complex knowledge into hierarchical representations of data with increasing levels of abstraction.

2.2 Deep Neural Networks

A cornerstone of the successes of deep learning is DNNs [6]. DNNs are composed of multiple nonlinear operations called layers. The essence of DNNs is that each layer transforms the input representations from the layer below into slightly more abstract representations. The first layer is referred to as the *input layer* which receives the observable raw data. Then, a series of layers are put on top of the input layer. Since these layers extract hierarchical abstract representations which are not observable from the raw data, they are referred to as the *hidden layers*. Finally, the last layer is referred to as the *output layer*. It computes the output predictions for given tasks from the last hidden representations.

One of the key aspects that differentiate one deep learning algorithm from another deep learning algorithm is the overall structure of DNNs. It is also often referred to as deep learning model architectures. They include a variety of architecture design considerations, such as which types of layers are used to construct DNNs and how they are connected with other layers in the model. In the following subsections, we will explain some of the representative DNN architectures by categorizing them into four groups: MLPs, CNNs, RNNs, and TFMs.

2.2.1 Multi-layer Perceptrons

MLPs are the most basic deep learning model architecture (Figure 2.2). They are also often referred to as feedforward neural networks since the information signal only flows from the input layer to the output layer and not vice versa. MLPs are composed of a single layer type called fully-connected layers. We can understand the MLPs as natural extensions of conventional linear regression models so that MLPs enable us to approximate more complex nonlinear functions.

The building block of fully-connected layers is computational units called

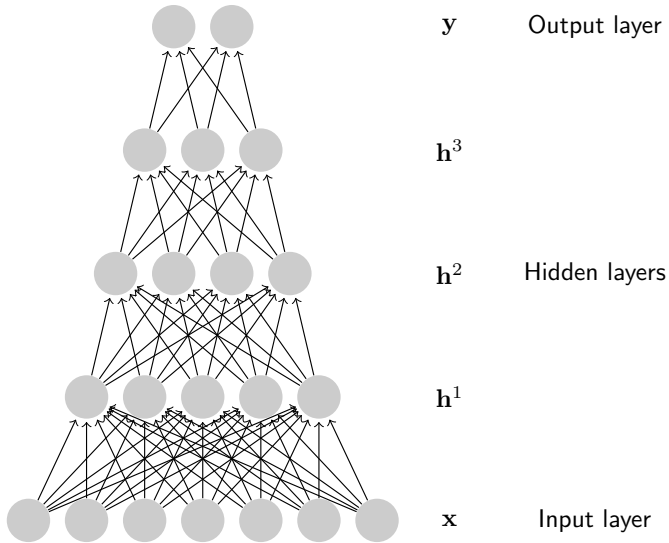

Figure 2.2: A diagram showing the basic structure of a MLP. Input layer is denoted as $\mathbf{x}$; hidden layers are denoted as $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$; output layer is denoted as $\mathbf{y}$. Each neuron in the hidden and output layers computes a weighted sum and a non-linear function of its input values.

neurons. Loosely inspired by neuroscience, each neuron receives a vector input from multiple neurons and computes a scalar output value using a possibly nonlinear activation function [5]. Let $\mathbf{h}^l$ denote a d^l-dimensional dense vector representations of l-th fully-connected layer, where $\mathbf{h}^0$ indicates the input data $\mathbf{x}$. Then, i-th neuron in the $(l+1)$-th fully-connected layer computes a scalar output value as

$$h_i^{l+1} = \sigma(\mathbf{w}_i^{l+1}\mathbf{h}^l + b_i^{l+1}) \tag{2.1}$$

where $\mathbf{w}_i^{l+1}$ and b_i^{l+1} are the weight vector and scalar bias, respectively. A variety of nonlinear functions can be used as the activation function σ including sigmoid, hyperbolic tangent, or rectified linear functions.

In conventional machine learning algorithms, MLPs have been used as a classifier such that hand-designed features are given as inputs. On the other hand, in representation learning algorithms, they are used to learn more informative hierarchical representations from raw data. Due to their capability

in analyzing high-dimensional data, MLPs are of great importance and often used as a component of more complex deep learning model architectures such as TFMs.

2.2.2 Convolutional Neural Networks

CNNs are also a type of feedforward neural networks. They are specially designed for data with a grid-like topology such as two-dimensional images or one-dimensional biological sequences. CNNs are directly inspired by the visual cortex of the human brain. In the visual cortex, there is a hierarchy of cell types *e.g.*, simple cells and complex cells [21]. First, the simple cells react to primitive patterns in sub-regions of visual stimuli. Then, the complex cells synthesize the information from simple cells to identify more intricate forms. Motivated by the powerful human vision system, CNNs leverage three essential ideas to capture better representations: sparse interactions, parameter sharing, and equivariant and invariant representations [5].

CNNs are composed of two types of layers: convolution and pooling layers (Figure 2.3). The key element of convolution layer is, of course, the convolution operation. Given a two-dimensional input function $I(i, j)$ and a two-dimensional weighting function $K(m, n)$, the convolution operation computes a weighted summation of inputs at every point as:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n). \tag{2.2}$$

Since the flipping of the weighting function does not hold much importance in terms of the machine learning perspectives, cross-correlations are more generally used for implementations of neural networks and simply called as the convolutions:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n). \tag{2.3}$$

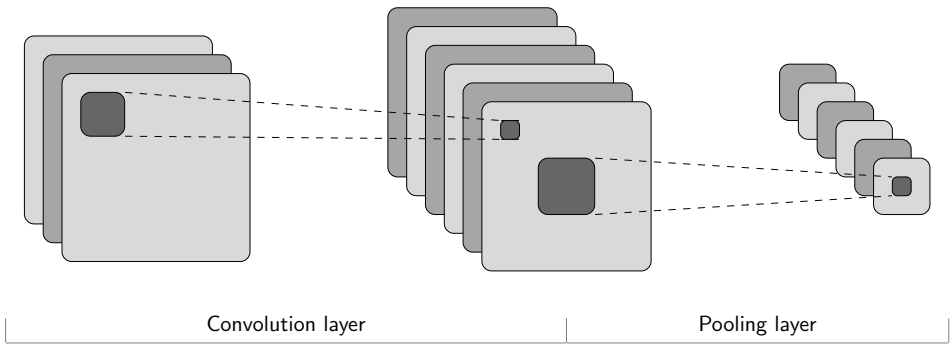

Figure 2.3: A diagram showing the basic structure of a CNN. Convolution layers discover local discriminative features with invariance to locations. Pooling layers aggregate statistics of local features enabling invariance to local transitions.

In CNNs, the weighting function $K(m, n)$ is usually referred to as the *kernel* or *filter*; the output $S(i, j)$ is usually referred to as the *feature maps*. Convolution layers hold multiple filters and additionally apply nonlinear activation functions for the feature maps.

We can understand the advantages of convolution layers more easily by comparing them with the fully-connected layers. Fully-connected layers have dense interactions with an input vector, where each output unit computes weighted summation of every input unit. On the other hand, convolution layers have sparse interactions. Each filter only computes the weighted summation of a small number of local input units. Thus, rather than looking at the entire image or sequences, it can more easily discover meaningful locally correlated patterns from data. Furthermore, convolution layers use the same set of shared filters regardless of the input locations within data. The parameter sharing strategy enables us to obtain location-equivariant representations and also improve training efficiency by significantly reducing the number of required weight parameters.

While convolution layers find distinctive patterns, pooling layers compute local summary statistics. For example, max-pooling layers compute the max-

imum value from sub-regions in feature maps and neglect the other values. In essence, pooling layers provide invariance to local transitions. Thus, they enable us to identify locally transformed but semantically similar features more robustly. In addition, since pooling layers generally summarize the feature maps into smaller sizes, they also reduce memory requirements and improve training efficiency.

CNNs have made significant advancements in computer vision tasks such as image recognition and object detection [6]. On top of the standard convolution and pooling layers, various architectural modifications (*e.g.*, residual connections [22] and squeeze-and-excitation modules [23]) are continuously developed to improve the performance of CNNs.

2.2.3 Recurrent Neural Networks

RNNs are specialized neural networks for analyzing sequential information [5]. Their biggest difference compared to the previous feedforward neural networks (*i.e.*, MLPs and CNNs) is that the hidden units have feedback connections. RNNs maintain *state vectors* in their hidden units. Then, they recurrently process an input sequence one element at a time. They update the state vectors based on the previous state vectors and the current input element of a sequence. The feedback connection allows the state vectors to be fed back into the hidden units. Since the state vectors are computed based on all the previous input elements, they implicitly store summarized information of all the previous elements.

Figure 2.4 shows the basic structure of a RNN. When we unroll them in time, we can more clearly understand their computations. At time step t, the hidden unit h_t receives input from the current input element x_t and itself from the previous time step h_{t-1} as:

$$\mathbf{h}_t = \sigma(\mathbf{w}_x \mathbf{x}_t + \mathbf{w}_h \mathbf{h}_{t-1} + \mathbf{b}), \tag{2.4}$$

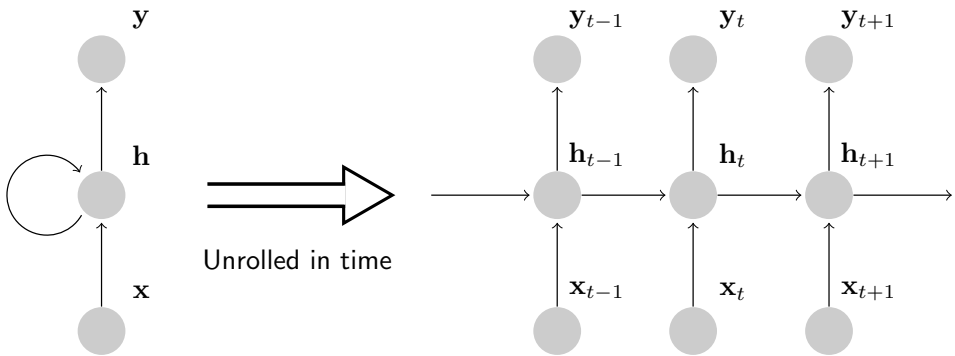

Figure 2.4: A diagram showing the basic structure of a RNN. At time step t, the hidden unit computes its state h_t based on its previous state h_{t-1} and current input x_t. The output o_t is computed from the h_t which depends on all the previous inputs $x_{t' \leq t}$.

where $\mathbf{w}_x$ and $\mathbf{w}_h$ are the weight vectors of the hidden unit for the input element and the previous state vector, respectively. Note that the weight vectors do not depend on the time index. As CNNs share weight parameters of filters across different locations within data, RNNs share weight parameters of hidden units across different time steps within data. It allows us to discover discriminative sequential patterns regardless of their locations within the sequence. In addition, parameter sharing reduces the required parameters by using the same weights for different time steps. It enables models to more robustly generalize to sequence data, which are even longer than those seen during the training.

Based on the basic structure (Figure 2.4), there are a variety of extensions of RNN architectures. For example, as in the feedforward neural networks, it is also common to use multiple hidden layers for RNNs. In addition, for problems where output may depend on both past and future input elements, bidirectional models have shown great results [6]. Bidirectional RNNs with

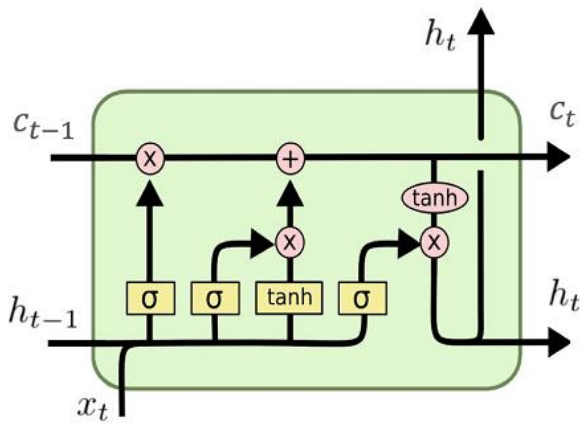

Figure 2.5: A diagram showing a LSTM unit [24]. It maintains cell states $\mathbf{c}_t$ as an explicit memory and controls the flow of information using multiplicative gates. This Figure is from Christopher Olah's blog [25].

L-layers can be defined as:

$$\begin{aligned}
\overrightarrow{\mathbf{h}_t^l} &= \sigma(\overrightarrow{\mathbf{w}_x^l}\mathbf{h}_t^{l-1} + \overrightarrow{\mathbf{w}_h^l}\mathbf{h}_{t-1}^l + \overrightarrow{\mathbf{b}^l}), \\
\overleftarrow{\mathbf{h}_t^l} &= \sigma(\overleftarrow{\mathbf{w}_x^l}\mathbf{h}_t^{l-1} + \overleftarrow{\mathbf{w}_h^l}\mathbf{h}_{t+1}^l + \overleftarrow{\mathbf{b}^l}), \\
\mathbf{h}_t^l &= \sigma(\mathbf{w}_h^l[\overrightarrow{\mathbf{h}_t^l}; \overleftarrow{\mathbf{h}_t^l}] + \mathbf{b}^l) \quad for \quad l = 1, \cdots, L,
\end{aligned} \tag{2.5}$$

where $\mathbf{h}_t^l$ denotes combined hidden states for l-th layer at time step t. Bidirectional RNNs have two sets of hidden states, each containing separate weight vectors. Forward hidden states $\overrightarrow{\mathbf{h}_t^l}$ are computed by processing the input in forward direction and contain summarized information of $x_{t' \leq t}$. Backward hidden states $\overleftarrow{\mathbf{h}_t^l}$ are computed by processing the input in reverse direction and contain summarized information of $x_{t' \geq t}$. Therefore, the combined hidden states can hold information for the whole input sequence.

Although RNNs are designed to analyze sequences with arbitrary lengths, vanilla models have showed difficulties in learning long-term dependencies. The same weights are multiplied recurrently, which makes gradients typically vanish if propagated over many time steps. This is called the vanishing gradient problem. One of the most effective solutions is using special hidden units

with explicit memory cells. Through self-loops within the hidden units, they produce paths for accumulating information and pass gradients for a long time [6]. The long short-term memory (LSTM) unit is the first of the kind and have effectively used in many practical applications [24]. It has a set of gating mechanism controlling the flow of information as:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{w}_{f,x}\mathbf{x}_t + \mathbf{w}_{f,h}\mathbf{h}_{t-1} + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{w}_{i,x}\mathbf{x}_t + \mathbf{w}_{i,h}\mathbf{h}_{t-1} + \mathbf{b}_i), \\
\mathbf{o}_t &= \sigma(\mathbf{w}_{o,x}\mathbf{x}_t + \mathbf{w}_{o,h}\mathbf{h}_{t-1} + \mathbf{b}_o), \\
\tilde{\mathbf{c}}_t &= \sigma(\mathbf{w}_{c,x}\mathbf{x}_t + \mathbf{w}_{c,h}\mathbf{h}_{t-1} + \mathbf{b}_c), \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \circ \sigma(\mathbf{c}_t),
\end{aligned} \tag{2.6}$$

where $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ denote the multiplicative forget, input, and output gates, respectively. The LSTM unit maintains cell states $\mathbf{c}_t$ as an explicit memory (Figure 2.5). Based on the contexts, the multiplicative gates are used to control the flow of information. The forget gate decides information to clear from the old cell states. The input gate decides information to update into the new cell states. The output gate decides information to deliver to the next layers. There are other variants of the LSTM units such as the gated recurrent units [26]. However, large-scale analyses have shown that no clear alternatives exist that can improve the performance significantly [27].

2.2.4 Transformers

TFMs are a type of feedforward neural networks proposed for sequence-to-sequence learning algorithms [28]. Before going into the details, we will explain the background of the attention mechanism, which is the core principle of TFMs. The objective of sequence-to-sequence learning algorithms is to transform a variable-length sequence from one domain to a variable-length

sequence from another domain. For example, in a neural machine translation problem, they aim to transform a sentence from one language into another language. Most sequence-to-sequence learning algorithms have an encoder-decoder model architecture [29]. As similar to autoencoders, an encoder converts the input sequence into representations called a context vector. Then, a decoder generates a transformed sequence from the context vector.

Since early sequence-to-sequence learning algorithms are based on RNNs, the encoder's last hidden states $\mathbf{h}_n$ are usually used as the context vector. However, one clear limitation is that the fixed-size context vector is often too small to summarize all the information within long sequences. Then, the attention mechanism was proposed to mitigate the limitation [30]. It uses attention scores to infer how strongly a model should attend to each input element for the generation of each output element. More specifically, rather than using a single context vector for generation of the whole output sequence, the attention mechanism builds a customized context vector $\mathbf{c}_t$ for each output element as:

$$\begin{aligned} \mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i, \\ \alpha_{t,i} &= \frac{\exp(\text{score}(\mathbf{y}_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\mathbf{y}_{t-1}, \mathbf{h}_{i'}))}, \\ \text{score}(\mathbf{y}_{t-1}, \mathbf{h}_i) &= \sigma(\mathbf{w}_a[\mathbf{y}_{t-1}; \mathbf{h}_i] + \mathbf{b}_a), \end{aligned} \tag{2.7}$$

where the attention scores are obtained from a jointly trained fully-connected layer with weight parameters $\mathbf{w}_a$. The context vectors are the sum of hidden states weighted by attention scores $\alpha_{t,i}$. Higher attention scores indicate that the hidden state $\mathbf{h}_i$ is more relevant for the generation of the next output $\mathbf{y}_t$. In the machine translation, the attention scores generally match the correspondence between words in source and target language.

While TFMs consist of various components (*e.g.*, fully-connect feedforward layers, residual connections, and normalization layers), the key component of TFMs is self-attention layers (Figure 2.6). The previous attention mechanism

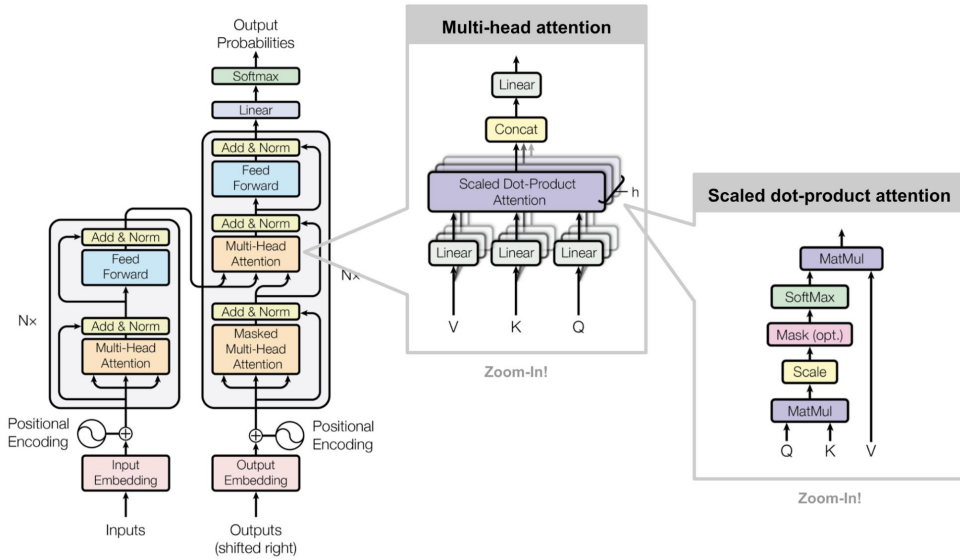

Figure 2.6: A diagram showing the basic structure of a TFM [28]. This Figure is from Lilian Weng's blog [25], which is a combined version of Figure 1 & 2 in the original paper.

only deals with the relationships between elements from two sequences. On the other hand, the self-attention mechanism, which is also referred to as the intra-attention mechanism, deals with the elements in a single sentence as well. For the encoder, it replaces the RNNs which are used to compute representations of the input sequence. Instead, the self-attention layers compute the representations of each input element by attending to other elements of the same sequence. Then, for the decoder, they attend to elements from both the input sequence and the generated output sequences as well. By directly analyzing the relationships of all the pairwise elements, the self-attention layers are more capable of capturing long-term dependencies within a sequence.

A self-attention layer is composed of multiple attention heads [28]. Given an input sequence, $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$, an attention head computes the output

representations, $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_n]$ as:

$$\begin{aligned} \mathbf{z}_i &= \sum_{j=1}^{n} \alpha_{ij}(\mathbf{x}_j \mathbf{w}_v), \\ \alpha_{ij} &= \frac{\exp(\mathbf{e}_{ij})}{\sum_{k=1}^{n} \exp(\mathbf{e}_{ik})}, \quad \mathbf{e}_{ij} = \frac{(\mathbf{x}_i \mathbf{w}_q)(\mathbf{x}_j \mathbf{w}_k)^T}{\sqrt{d_z}}. \end{aligned} \tag{2.8}$$

Each output element is a weighted sum of *value vectors* computed by weight parameters $\mathbf{w}_v$. Each attention coefficient, α_{ij}, is computed by taking dot products of the *query vector* of the current element with the *key vector* of other elements computed with weight parameters $\mathbf{w}_q$ and $\mathbf{w}_k$, respectively.

2.3 Training of Deep Neural Networks

Once we have defined deep learning model architectures, the next important step is to train the DNNs. The goal of training is to optimize the weight parameters in each layer so that the most suitable hierarchical representations can be learned. Currently, most deep learning algorithms adopt gradient-based iterative optimization method called *gradient descent* for their training [5]. Suppose we have a DNN f where θ denotes its weight parameters. The training of f is done by repeating the following processes with training data $(\mathbf{x}, \mathbf{y})$. First, we use $\mathbf{x}$ as inputs to the DNN. It sequentially computes the representations in hidden layers and produces prediction outputs $\tilde{\mathbf{y}}$. This process is referred to as the *forward propagation*. Then, we use a pre-defined objective function to compute training loss $J(\theta) = L(\mathbf{y}, \tilde{\mathbf{y}})$ between the true outputs and the prediction outputs. The objective function is also referred to as the loss function or the error function. The weight parameters are updated with the derivative of the objective function with regard to the weight parameters as:

$$\theta \longleftarrow \theta - \epsilon \nabla_\theta J(\theta). \tag{2.9}$$

In order to compute the gradients with regard to all the weight parameters, the error signal is propagated backward through the DNN using the chain rule. This process is referred to as the *backward propagation* or *back-propagation*. The gradients of the objective function indicate its slope in the parameter space. Thus, by iteratively modifying θ in small steps towards the opposite direction of the gradients, we can minimize the objective function step by step. While we can use the entire training data to compute gradients for every parameter update, in practice, deep learning algorithms generally adopt *stochastic gradient descent* which uses a minibatch of uniformly sampled examples to more efficiently estimate the gradients.

To accelerate the training of DNNs, numerous variants of stochastic gradi-

ent descent with additional mechanisms have been developed. For example, momentum methods accumulate gradients with an exponentially decaying moving average to more continuously move in one direction [31]. They are shown to accelerate learning, especially when gradients are noisy, small but consistent, and have high curvature [5]. Meanwhile, *RMSProp* uses adaptive per-parameter learning rates to accommodate sensitivity differences of each parameters [32]. *Adam* combines both approaches and applies per-parameter learning rates from the first-order and second-order moments of the gradients [33]. While the gradient-based optimization methods vary in detail, they still share the learning rate as one of the hyperparameters. Various learning rate schedulers are used to gradually decrease the learning rates over time, including step decay, exponential decay, and recently proposed cosine annealing with warm restarts [34].

There are still numerous other factors that can significantly affect the training of DNNs. One factor is weight initialization. A common idea is to randomly initialize them using Gaussian distribution with zero mean. However, if the variance is set too small or large, it quickly makes all activations become zero or one, respectively. In both cases, gradients become too small for stable training. *He initialization* strategy provides a solution by adjusting variance based on the number of weights connected to each output unit [35]. Another factor is internal covariate shift, which indicates the change of input distributions of hidden layers due to the simultaneous updates of all the layers. It often causes insatiability of training and requires lower learning rates. *Batch normalization* alleviates the problem by forcing inputs of hidden layers to have a unit Gaussian distribution at the beginning of training [36]. It consists of two operations: (1) normalization of each input based on its mean and variance within a minibatch, and (2) linear transformation of each input to restore the representation power of the network.

Another essential element in the training of DNNs is how to avoid overfit-

ting. DNNs are often too complex and trained to fit noises as well, rather than focusing only on the relevant signals. It hurts the generalization performance of models leading to lower training error but higher test error. Regularization strategies put extra constraints on model parameters or the objective function to lead to a simpler model. Norm penalties and dropout are the two most widely used regularization strategies. Norm penalties limit the model capacity by adding a l_1 or l_2 to the objective function. Dropout randomly sets some hidden units to zero so that the model cannot heavily rely on any single representations [37].

2.4 Representation Learning in Bioinformatics

A variety of representation learning methods have been widely used in various bioinformatics domains (Figure 2.7). Researchers have used different neural network architectures (*i.e.*, MLPs, CNNs, RNNs, and TFMs) based on their research objectives and corresponding data characteristics. In this section, we will briefly review previous works which used representation learning methods in bioinformatics problems and focus on the strengths of each model architecture. We note that some contents of this section are referenced from our review paper [7].

MLPs have strengths in analyzing high-dimensional data and discover their complex relationships. In early works, MLPs were used more as a classifier with pre-processed hand-crafted features as inputs. For example, expression measurements of landmark genes were used as inputs to infer the expression of target genes [38]. Frequency components of electroencephalography (EEG) signals were used as inputs to classify left-hand and right-hand motor imagery skills [39]. More recently, MLPs were often used as a component of deep autoencoders to learn effective highly-abstract representations. Iterative training of deep autoencoders improved prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins [40]. Deep autoencoders were also used to learn high-level representations from breast cancer histopathology images and detect image patches containing nuclei. With a more advanced training method for imbalanced data and a DNA-specific regularization technique, MLPs also showed great ability for predicting splice junctions from DNA sequences and detecting subtle non-canonical splicing signals [41].

CNNs have strengths in discovering patterns equivariant to locations and invariant to small transformations. Since they were originally proposed for analyzing general images, numerous works proposed to analyze biomedical images using CNNs. FingerNet proposed a finger joint detection system, which

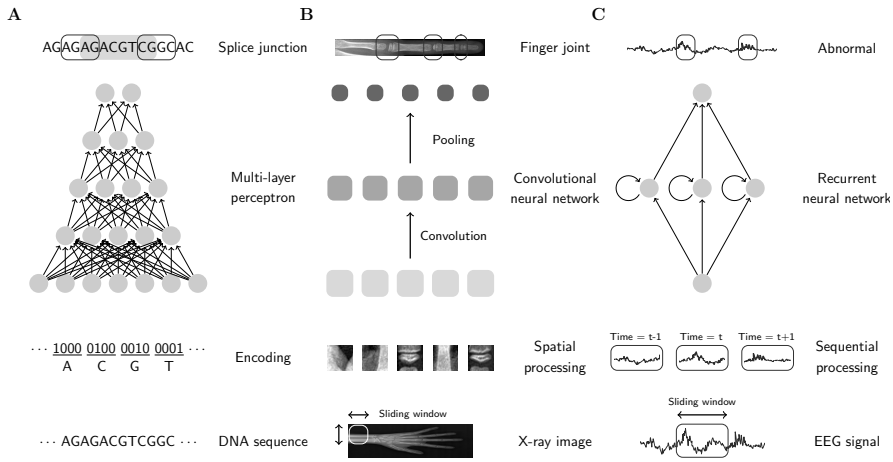

Figure 2.7: Research examples of representation learning in bioinformatics. (A) Prediction of a splice junction from DNA sequences with a MLP [41]. (B) Finger joint detection from X-ray images with a CNN [42]. (C) Abnormal detection from EEG signals with a RNN [43].

is a crucial step for medical examinations of bone age, growth disorders, and rheumatoid arthritis [42]. A cascaded CNN architecture was also proposed to segment brain tumors from magnetic resonance images by exploiting both local and global contextual features [44]. On the other hand, the strengths of CNNs have also made great advances in one-dimensional data such as biological sequences. In particular, for genomic sequence analysis, they have a significant similarity with the traditional approaches. Traditional approaches often incorporate hard-coded position-specific scoring matrices to identify regulatory motifs. The filters in convolution layers can be understood as learnable position-specific scoring matrices. Furthermore, they enable discovering more complex and longer motifs, integrate cumulative effects of observed motifs, and eventually learn more sophisticated regulatory codes [45]. For example, DeepBind [46] and Basset [47] proposed CNN models for transcription factor binding site prediction and 164 cell-specific DNA accessibility multitask prediction, respectively. DeepSEA proposed multitask joint learning of chromatin factors (*i.e.*, transcription factor binding, DNase I sensitivity, histone-mark

profile) and prioritized expression quantitative trait loci and disease-associated genetic variants [48].

RNNs have strengths in analyzing sequential information and handling variable-length sequence data. Thus, for biological sequence data, they have been a natural choice of DNN architectures. While one of the early works used RNNs with perceptron hidden units for protein secondary structure prediction [49], most recent works used multiplicative hidden units such as LSTM units or gated recurrent units. DeepMirGene [50] and DeepTarget [51] used the LSTM units for microRNA identification and target prediction and significantly outperformed the SOTA approaches. Furthermore, RNNs are often combined with CNNs to better capture local discriminative patterns. DeepLoc proposed a convolutional RNN with the long-short term memory units for subcellular localization of protein sequences [52]. A similar hybrid neural network was also proposed for sleep stage classification with time and frequency components of pediatric scalp EEG recordings [43].

TFMs have strengths in capturing long-term dependencies within data. Since TFMs were proposed most recently, only a couple of works have used them for bioinformatics. Inspired by BERT (Bidirectional Encoder Representations from TFMs), BioBERT proposed a pre-trained domain-specific language representation model for biomedical text mining [53]. By pre-training a TFM on large-scale biomedical corpora, it showed significant performance improvement in biomedical named entity recognition, biomedical relation extraction, and biomedical question answering. In addition, TFM models have been used for protein sequences as well. TALE proposed a TFM-based model to annotate protein function with the joint embedding of gene ontology labels [54]. TransformerCPI improved the prediction of protein interaction with chemical compounds, which is essential for drug discovery and chemogenomics research [55].

2.5 Biological Sequence Data Analyses

A sequence can be defined as a series of elements where their orderings as well as identities of each element contain critical information. As a matter of fact, a lot of data around us are actually sequence data. Texts are sequences of words, videos are sequences of images, and stock market data are sequences of time-varying prices. Each word, image, and price hold essential information, but the change of their orderings can make completely different information. For example, think about two sentences: (1) "He can even speak English" and (2) "Even He can speak English." The two sentences are composed of the same five words, but they carry entirely different meanings due to the different orderings of words. The former implies that the fact is surprising, and the latter implies English is easy to learn.

A lot of biological information is carried in the form of sequence data. DNA and protein sequences have genetic information and are composed of a series of nucleotides and amino acids, respectively. As in the previous examples, both identities of each element and their orderings in biological sequence data contain critical information. One simple example is codons. Genetic information in DNAs is transferred to RNAs through transcription and to proteins through translation. During the processes, three consecutive nucleotides called codons encode corresponding amino acids. According to the identities and orderings of nucleotides, there are 64 different codons where 61 codons encode 20 amino acids, and the remaining three serve as the stop signals [56].

There are numerous challenges that make it more difficult to analyze biological sequence data. First, biological sequences often have variable lengths. For instance, the number of nucleotides in a gene can vary. The variable-length issue often complicates the analyses by requiring a model to process and generalize regardless of their lengths. One of the conventional approaches used to circumvent the issue is feature extraction into fixed-size vectors. While distribution of k consecutive elements, called k-mer, is widely adopted, it can

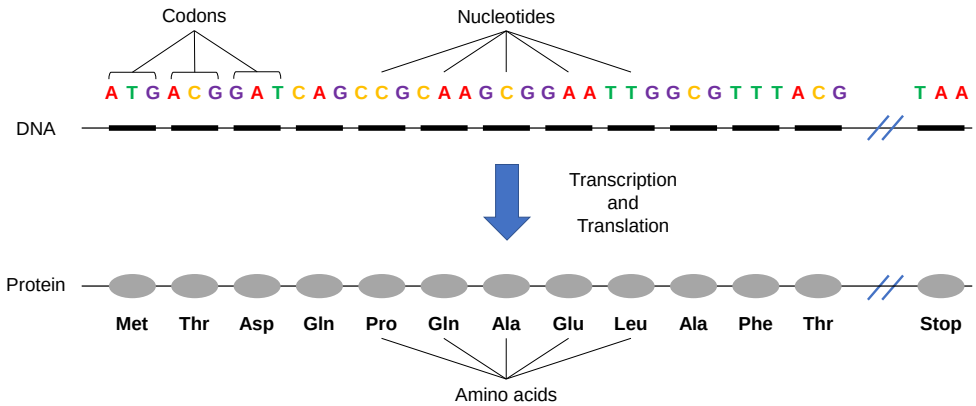

Figure 2.8: Illustration of codons. Three consecutive nucleotides of DNAs undergo transcription and translation to amino acids of proteins. This figure is redrawn from the one from National Human Genome Research Institute webpage [56].

only capture the dependencies within k. In representation learning methods, it is more common to truncate or pad sequences to a pre-defined length. Some models also support the handling of variable-length data such as RNNs and CNNs with a global max-pooling layer [7].

Another challenge is capturing both short-term and long-term dependencies. Biological sequences are often extremely long, and elements in distant positions can also have close dependencies. For example, proteins naturally fold into three-dimensional structures determined by their amino acid sequences. As these structures have a direct impact on protein functions, amino acids in distant positions often co-mutate to maintain the indispensable structures [57]. Each DNN architecture holds different capabilities in capturing short-term and long-term dependencies. While CNNs and RNNs are more capable of capturing local patterns, recently proposed TFMs are better at capture extremely long-term dependencies by handling all pair-wise relationships equally [28].

To analyze biological sequence data with representation learning methods, it is necessary to pre-process and convert sequences into numerical vectors

[58]. In the following, we will explain the pre-process procedures step-by-step. The first step is tokenization which divides sequences into discrete segments. While it is common to divide them into each constituent element, dividing them into overlapping k-mers are also often adopted for some works [59]. The second step is vocabulary construction. Considering all possible elements can sometimes significantly increase the size of input vectors. For example, if we use overlapping k-mers, the number of possible elements increases exponentially with the length k. Thus, vocabulary construction limits the number of elements to consider and handles the other elements as the same token. The third step is variable-length handling. It truncates or pads variable-length sequences to a pre-defined length. The last step is encoding. Most methods use one-hot encoding. It converts an item into a binary vector, in which all the values are set to 0 except for the one corresponding to the element is set to 1.

2.6 Evaluation Metrics

In this section, we review both classification and regression evaluation metrics used in the contents of the dissertation. First, for a binary classification problem, we can categorize a prediction result into four categories based on the class of a target and the correctness of the prediction (Figure 2.9). True positive (TP) and true negative (TN) indicate correctly predicted samples for positive and negative targets, respectively. False positive (FP) and false negative (FN) indicate incorrectly predicted samples for positive and negative targets, respectively. Then, numerous evaluation metrics are defined based on the categorization. The most common and familiar classification metric would be accuracy defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.10}$$

While accuracy can show the overall correctness of classification models, it does not take account of the correctness for identifying each class. This limitation is more critical for imbalanced datasets. Suppose we have an inactive classifier that always outputs negative predictions and a dataset composed of 97% of negative targets and 3% of positive targets. Then, the accuracy of the classifier is 97% even if it cannot correctly classify any of the positive targets.

Several evaluation metrics are used to quantify the performance for identifying each class. Sensitivity and specificity measures the fraction of correct predictions for each class as:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{2.11}$$

Sensitivity is also referred to as recall and true positive rate. It shows how well a classifier can correctly classify samples from positive classes. Specificity is also referred to as true negative rate and shows how well a classifier can

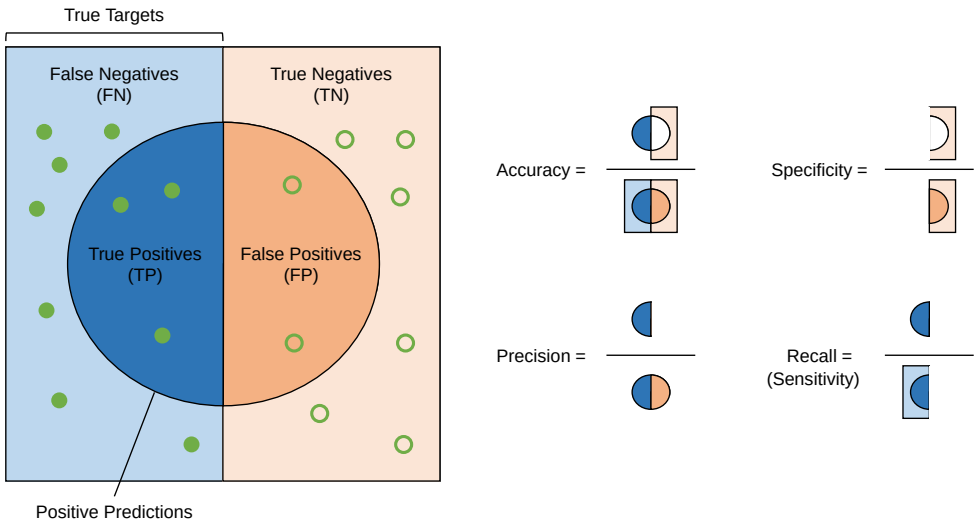

Figure 2.9: An illustration of classification evaluation metrics, *i.e.*, accuracy, specificity, precision, and recall (sensitivity).

correctly classify samples from negative class. While sensitivity and specificity are defined in terms of the true class of a target, precision measures the quality of positive predictions as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.12}$$

Precision shows the proportion of correctly classified samples among all the positive predictions. High precision indicates that the classifier is trustful when it identifies a sample as a positive class. Finally, F_β is defined by combining precision and recall as:

$$\text{F}_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}. \tag{2.13}$$

As similar to accuracy, F_β score shows the overall classification performance of a classifier but also takes account for the correctness for identifying each class. If β is set to 1, it is simply the harmonic mean of precision and recall. If β is larger than 1, F_β score considers recall as β times more important as

precision.

Most classifier models produce prediction scores in real value. Then, various thresholds are used to binarize them afterward. The previous metrics can only evaluate the binarized predictions and cannot assess the prediction scores in terms of different thresholds. The area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPRC) are the two classification evaluation metrics to mitigate the limitation. AUROC computes the area under a line plot of the true positive rate (sensitivity) against the false positive rate (1-specificity) calculated at various thresholds. AUPR computes the area under a line plot of the precision against the recall computed at various thresholds. Higher AUROC and AUPR generally indicate that a classifier would produce higher predictions for positive instances than negative instances. They provide more robust ways to evaluate classifiers without selecting explicit decision thresholds.

The most straightforward regression evaluation metrics are computing the errors between the true targets and prediction scores. For example, mean absolute error (MAE) measures absolute average errors and mean squared error (MSE) measures squared average errors as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - p_i|}{n}, \quad \text{MSE} = \frac{\sum_{i=1}^{n} (y_i - p_i)^2}{n}, \tag{2.14}$$

where y_i and p_i denote the true target and prediction score of a sample, respectively. MSE is more generally used than MAE for tasks where large errors should be particularly more penalized. Additionally, correlations are also widely used to evaluate regression models. Pearson correlation r is used to show the strength of linear relationships between the true targets and prediction scores as:

$$r = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^{n} (p_i - \bar{p})^2}}, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \quad \bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i. \tag{2.15}$$

Although Pearson correlation is useful for most regression problems, it can be problematic if the distribution of true targets is changed. For example, in CRISPR-Cpf1 activity prediction (Chapter 3), true activity distributions can significantly vary for each batch of experiments due to laboratory conditions, personnel differences, and many other factors [60]. This is called a batch effect. In such cases, Spearman correlation ρ is a more robust regression evaluation metric showing the strength of monotonic relationships between the true targets and prediction scores. It measures the Pearson correlation between rankings of the true targets and prediction scores rather than their raw values.

Chapter 3

CRISPR-Cpf1 Activity Prediction

Targeted genome editing using CRISPR system has rapidly become a mainstream method in molecular biology [12, 61]. Cpf1 (from Prevotella and Francisella 1), a recently reported effector endonuclease protein of the class 2 CRISPR system, has a few different characteristics from the predominant Cas9 nuclease. Although Cpf1 has broadened our options to efficiently modify genes in various species and cell types, we still have limited knowledge on Cpf1, especially regarding its target sequence-dependent activity profiles [62, 63].

CRISPR is an innate adaptable immune mechanism of bacteria, of which we now take advantage for efficient targeted genome editing. There are two primary components of the CRISPR system. The first one is an endonuclease which cuts the target sequences, and the second one is a guide RNA which directs the endonuclease to them [9]. The CRISPR system enables us to delete, insert or replace DNAs in the genome. However, actually, we cannot cut any sequence in the genome. CRISPR has one constraint where it can be deployed: it can only cut the target sequences next to a short motif called PAM [64]. This is one of the reasons why researchers have searched for different endonucleases having distinct PAM sequences. Currently, Cas9 and Cpf1 are the most widely

used ones. In this chapter, we focus on Cpf1 with TTTV PAM sequence [65].

One of the biggest challenges of CRISPR technology is identifying exactly which part of a gene to target when a researcher wants to edit the gene. There can be thousands of potential target sites with PAM sequences. The problem is different target sites can have different CRISPR efficiencies. Some of them are more editable than others, but some of them rarely respond to the endonucleases. Therefore, choosing a target to rectify aberrant gene sequences with maximum efficiency has been a significant concern for researchers from across the world [10]. There are two critical genetic and epigenetic factors that are known to affect the CRISPR activity [11]. The genetic factor is a target sequence composition. The composition of a target sequence (*i.e.*, the upstream of the PAM sequence, the PAM sequence, protospacer which hybridizes with the guide RNA, and the downstream of the protospacer) dictates its thermodynamics and affects the CRISPR activity. The epigenetic factor is chromatin accessibility. Generally, if the target is more accessible, it shows higher activity since the CRISPR complex can approach the target more easily.

To date, various researchers have manually tested synthetic targets in lab-based experiments, but very few people have the expertise, time, or budget for the laborious work [66]. Several computational approaches have been proposed for the in silico prediction of CRISPR nuclease activities. However, they heavily relied on manual feature extraction, which inevitably limits the efficiency, robustness, and generalization performance [12, 11]. To address the limitations of existing approaches, this chapter presents an end-to-end deep learning framework for CRISPR-Cpf1 guide RNA activity prediction, dubbed as DeepCpf1. It incorporates (1) a CNN for feature learning from target sequence composition and (2) multi-modal architecture for seamless integration of an epigenetic factor (*i.e.*, chromatin accessibility).

To evaluate the prediction performance of DeepCpf1, we compared its performance with conventional machine learning algorithms, which are current

SOTA approaches for Cas9 activity prediction. When evaluated using Cpf1 activities at endogenous sites in two different cell lines (*i.e.*, HEK293T and HCT116), the proposed algorithm significantly outperformed the other approaches, reaching Spearman correlations of 0.87 and 0.77, respectively. We also evaluated it with other published datasets of different studies from independent laboratories and confirmed excellent generalization performance.

3.1 Methods

3.1.1 Model Architecture

DeepCpf1 receives a 34-nucleotide target sequence as input, and it produces a regression score that highly correlates with CRISPR-Cpf1 activity. In contrast to previous approaches [12, 11] that relied heavily on hand-crafted features (*e.g.*k-mer counts, melting temperature, and free energy), DeepCpf1 eliminates the need for laborious manual feature engineering, leveraged by the use of a CNN. DeepCpf1 can thus automatically learn informative representations of target sequences relevant to CRISPR-Cpf1 activity profiles.

DeepCpf1 was implemented using Theano [67] and Keras (http://keras.io) libraries. Figure 3.1 shows an overview of DeepCpf1, which proceeds in five stages. (1) The one-hot encoding input layer converts the sequence into numerical representations for downstream processing. It encodes the nucleotide in each position as a 4-dimensional binary vector, in which each element represents the type of nucleotide: A, C, G, and T. The encoding layer then concatenates the binary vectors into a 4-by-34 dimensional binary matrix representing the 34-nucleotide target sequence. (2) The convolution layer performs one-dimensional convolution operations with 80 filters of length 5. The filters slide along only one axis (*i.e.*sequence length) of the one-hot encoded matrix containing the four nucleotide channels. This process is equivalent to scanning learned PWMs across the target sequence in conventional techniques. The convolution layer then applies the rectified linear unit (ReLU) non-linear function [f(x)=max(0,x)] to the convolution outputs. The pooling layer computes the average in each of the non-overlapping windows of size 2, providing invariance to local shifts and reducing the number of parameters. (3) DeepCpf1 uses three fully connected layers with 80, 40, and 40 units, respectively. Each unit in the fully connected layers performs linear transformations of the previous layer's outputs and applies the ReLU non-linear function. Multiple non-linear

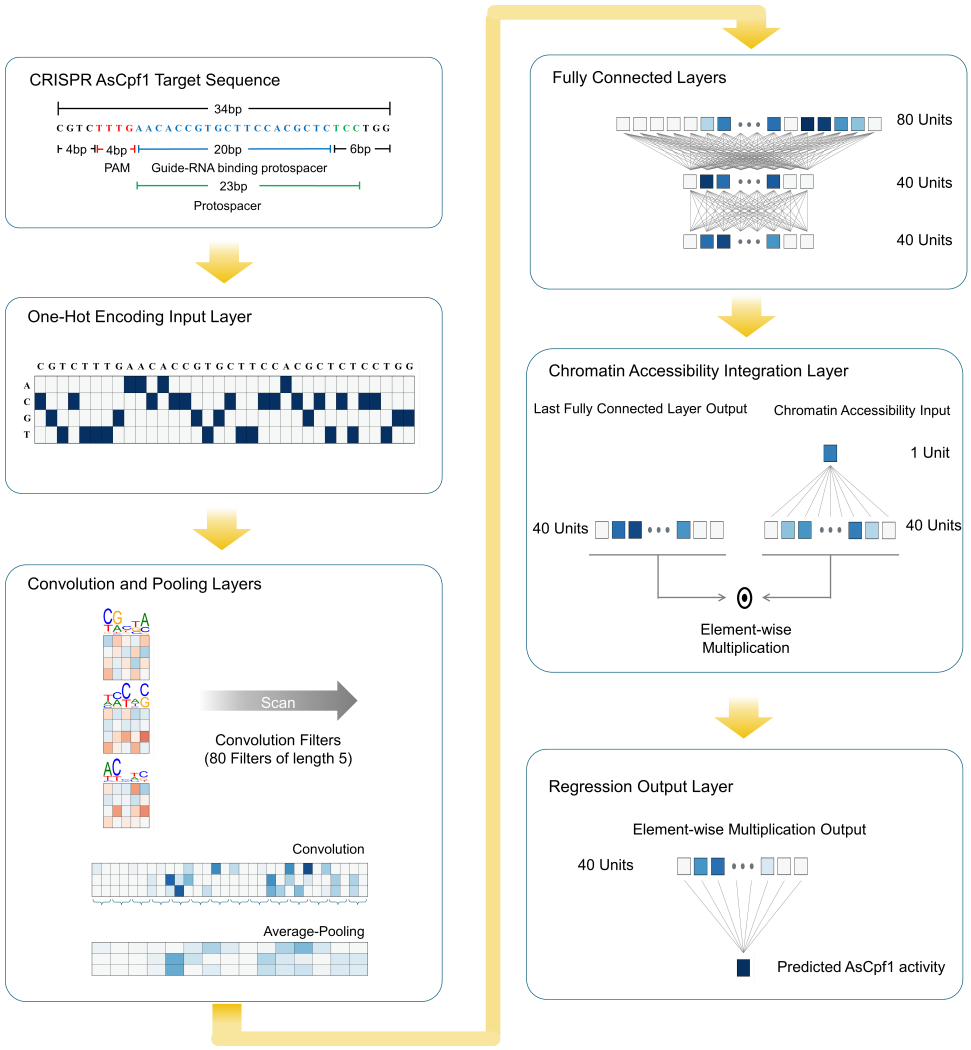

Figure 3.1: Overview of DeepCpf1.

layers enable the model to learn hierarchical representations of data with increasing levels of abstraction. (4) The chromatin accessibility integration layer incorporates the sequence representations with the chromatin accessibility information of the target sequence. It uses another fully connected layer with 40 units to transform the 1-dimensional binary chromatin accessibility input to a 40-dimensional real vector, which matches the output shape of the last fully connected layer. Then, it integrates the 40-dimensional chromatin accessibility and sequence representation vectors by performing element-wise multiplication. (5) The last stage, the regression output layer, performs a linear transformation of the outputs of the chromatin accessibility integration layer and predicts CRISPR-Cpf1 activity.

3.1.2 Training of Seq-deepCpf1 and DeepCpf1

We trained the proposed model in two main steps: (1) Model selection and pre-training of the entire architecture (we denote the model that has been trained up to this step as Seq-deepCpf1) with integrated target data, and (2) Fine-tuning with endogenous target data considering an additional chromatin accessibility input, which led to the development of DeepCpf1 (the outcome of the training process). In both steps, we optimized the mean squared error loss function using the Adam optimizer [33] and used dropout [37] for the model regularization with a 0.3 dropout rate.

First, we conducted nested cross-validation (CV) to demonstrate the reliability of the model selection (Figure 3.2). In each fold of the outer 10-fold CV, we randomly constructed training datasets with different sizes (*i.e.*n = 1,000, 2,000, 4,000, 8,000, and 13,500) to evaluate the performance improvements associated with different sizes of training datasets. Each training dataset was used for the following model selection in the inner 5-fold CV and training of the selected model. Of note is that the validation dataset (n = 1,500) was fixed for all of the training datasets of different sizes within the same fold of

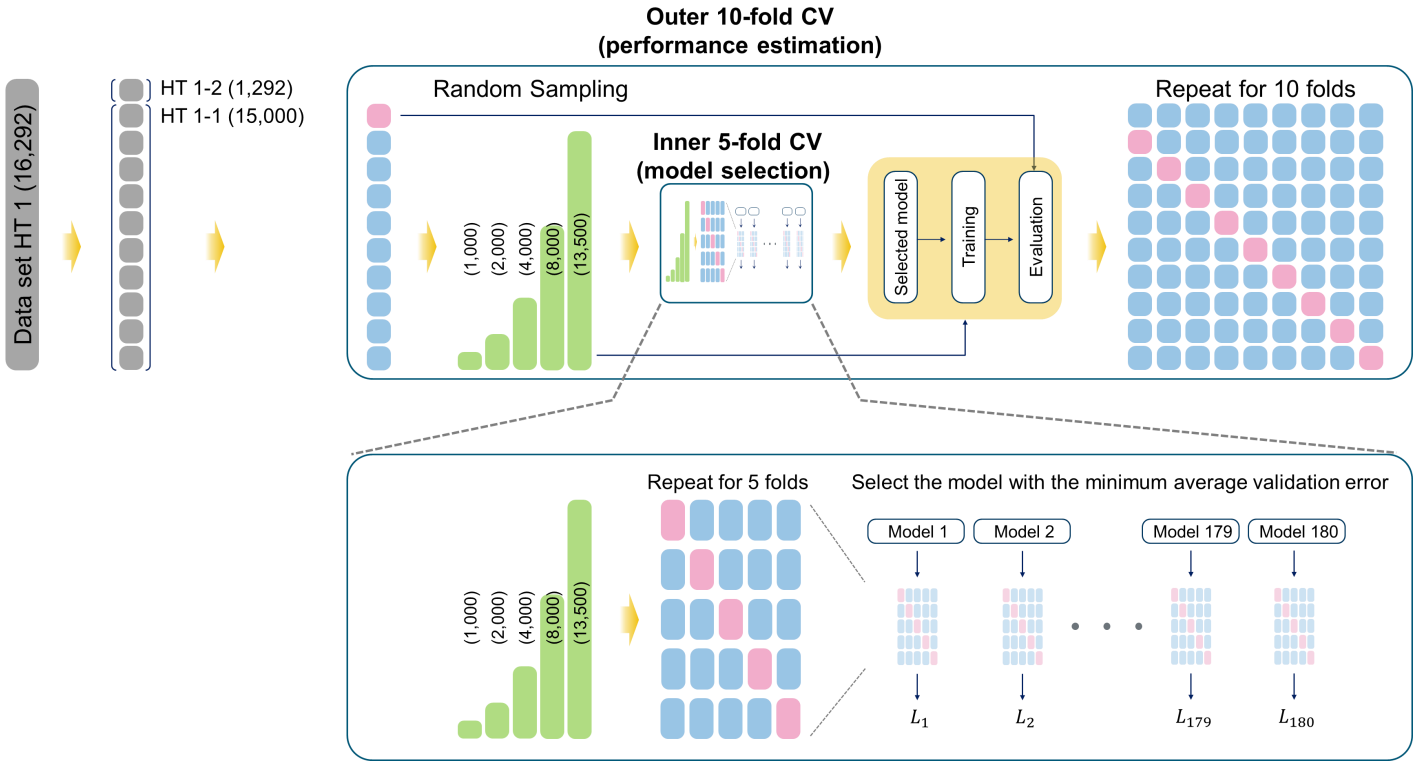

Figure 3.2: Diagram of nested cross-validation (CV). In the inner loop of this procedure, we performed 5-fold CV to set the values of model hyperparameters. In the outer loop, we performed 10-fold CV to train and validate the model selected from the inner loop of the nested CV procedure.

the outer CV. In each fold of the inner CV, the respective training and validation datasets were used to train and validate 180 model candidates with different hyperparameter configurations of the number of filters, filter lengths, the number of fully connected layers, and the number of units in each fully connected layer.

After verifying the expected performance of the proposed model, we carried out the final model selection using a 5-fold CV. Among the 180 model candidates with different hyperparameter configurations, we selected the model that showed the minimum average validation loss as the final model for Seq-deepCpf1. We then pre-trained the final model (102,681 free parameters), learning informative representations of target sequences relevant to CRISPR-Cpf1 activity profiles.

To take chromatin accessibility as well as the learned sequence representations into account, we adopted an additional chromatin accessibility integration layer right before the regression output layer. During the fine-tuning, we then optimized the mean squared error loss function, only updating the weight parameters in the last two layers (121 free parameters). By fixing the weight parameters in the other layers, DeepCpf1 could avoid overfitting and effectively learn to incorporate the sequence representations with the chromatin accessibility information.

3.2 Experiment Results

3.2.1 Datasets

Integrated datasets

We first obtained large-scale datasets of Cpf1 activity at 16,292 (experiment A) and 2,963 (experiment B) lentivirally integrated target sequences using a high-throughput method in HEK293T cells [11]. The high-throughput experiments A and B led to the generation of datasets HT 1 and HT 2, respectively, consisting of target sequence compositions and corresponding indel frequencies. dataset HT 1 was split into datasets HT 1-1 (n = 15,000) and HT 1-2 (n = 1,292) by random sampling. Note that the integrated target datasets are barely influenced by chromatin accessibility, thus can only be used for learning the effects of target sequence composition.

For high-throughput experiments A and B, a total of 67,301 Cpf1 target sequences (55,003 and 12,298 for experiments A and B, respectively) were designed from the coding sequences of 19,565 human genes using Cpf1-Database (http://www.rgenome.net/cpf1-database). We selected three or four Cpf1 target sequences from each gene in the majority of cases; the numbers of selected target sequences per gene were one in 927 genes, two in 893 genes, three in 6,461 genes, four in 11,255 genes, five in seven genes, six in nine genes, seven in eight genes, and eight in five genes. Instead of 23-nucleotide guide sequences, we used 20nt guide sequences because this guide RNA truncation perfectly preserves Cpf1 activity and 3' distal nucleotides outside of the 20-nucleotide crRNA guide sequence do not form heteroduplexes with target DNA [68]. We designed each oligonucleotide to contain the 20-nucleotide guide-RNA-encoding sequence, 20-nucleotide barcode, and 34-nucleotide target sequence in a total length of 130 nucleotides. For experimental convenience, the 67,301 oligonucleotides were arbitrarily divided into six groups (five groups for high-throughput experiment A and one group for high-throughput experiment B),

Table 3.1: Datasets for CRISPR-Cpf activity experiments.

Dataset	Type	Method	Cell line	Samples
HT 1-1	Integrated	High-throughput experiment A	HEK293T	15,000
HT 1-2	Integrated	High-throughput experiment A	HEK293T	1,292
HT 2	Integrated	High-throughput experiment B	HEK293T	2,963
HT 3	Integrated	High-throughput experiment C	HEK293T	1,251
HEK-lenti	Endogenous	Lentiviral transduction	HEK293T	148
HEK-plasmid	Endogenous	Plasmid transfection	HEK293T	55
HCT-plasmid	Endogenous	Plasmid transfection	HCT116	66
Kleinstiver 2016	Endogenous	Plasmid transfection	U2OS	22
Chari 2017	Endogenous	Plasmid transfection	HEK293T	18
Kim 2016	Endogenous	Plasmid transfection	HEK293T	10

datasets HT 1 to 3 contained the indel frequencies at integrated target sequences and were obtained from three independent high-throughput experiments conducted in HEK293T cells. The dataset from high-throughput experiment A was divided into datasets HT 1-1 and HT 1-2 by random sampling. Indel frequencies at endogenous human coding and non-coding regions were included in datasets HEK-lenti, HEK-plasmid, and HCT-plasmid. In the case of the experiment for dataset HEK-lenti, indel frequencies were evaluated at both the integrated target sequences and the corresponding endogenous target sites. Different delivery methods and cell lines were used to generate datasets HEK-lenti, HEK-plasmid, and HCT-plasmid and are described in the table.

which were independently synthesized by CustomArray, Inc (Bothell, WA). For the experiments in which datasets of indel frequencies at endogenous target sites were built, 223 oligonucleotides (148 for lentiviral transduction and 75 for transient transfection) containing guide RNA-encoding sequences, barcodes, and target sequences were synthesized by Cellemics, Inc (Seoul, South Korea) or Macrogen, Inc (Seoul, South Korea).

Endogenous datasets

We prepared three independent datasets of Cpf1 activities at endogenous sites in two different cell lines, *i.e.*HEK293T and HCT116 (datasets HEK-lenti, HEK-plasmid, and HCT-plasmid). Note that the endogenous target datasets are smaller and noisier than the integrated target datasets. They are significantly affected by chromatin accessibility.

To generate the dataset covering a wide range of DNase I sensitivities, 141 endogenous target sites were arbitrarily selected from genome regions with various DNase I sensitivities. The remaining 82 target sites were derived from four arbitrarily selected genomic regions within chromosomes 9, 15, 19, and 228. Indel frequencies at a total of 223 (= 141 + 82) endogenous sites were analyzed after transfection of plasmids encoding Cpf1 and crRNA (n = 55 for HEK293T cells and n = 66 for HCT116 cells (ATCC); 46 target sequences were shared by both experiments) or transduction of lentivirus encoding Cpf1 and crRNA (n = 148 for HEK293T cells). For the transfection, plasmids encoding Cpf1 (100ng) and crRNA (100ng) were delivered to 70 - 80% confluent HEK293T or HCT116 cells on a 96-well plate via Lipofectamine 2000 (Invitrogen). The next day, the culture medium was exchanged with DMEM supplemented with 10% FBS and 2 μg/ml of puromycin. Five days post-transfection, cells were harvested and analyzed for deep sequencing. For the transduction, HEK293T cells were seeded onto 48-well plates and infected with individual lentiviral vectors encoding crRNA and target sequence pairs. After three days of trans-

duction, cells were treated with 2 µg/ml puromycin for the following three to five days to select against untransduced cells. Next, cells were infected with Cpf1-encoding lentivirus, and cells were further selected with 20 µg/ml blasticidin S (InvivoGen, San Diego, CA). Five days after Cpf1-encoding lentivirus transduction, genomic DNA was isolated from the cells and subjected to deep sequencing.

We obtained binary chromatin accessibility information for these cell lines using DNase-seq narrow peak data from the Encyclopedia of DNA elements [69]. For each target site, 27 bases of PAM plus protospacer sequence were aligned to the hg19 human reference genome using bowtie [70]. Only the target sites that overlapped with DNase-seq narrow peaks were considered as DNase I hypersensitive target sites.

3.2.2 Baselines

To evaluate DeepCpf1, we compared its prediction performance with that of the following learning models that previously showed SOTA Cas9 activity prediction [12], *i.e.*L1-regularized linear regression, L2-regularized linear regression, L1L2-regularized linear regression, and gradient-boosted regression tree (Boosted RT), and an algorithm that was previously used for Cpf1 activity prediction, *i.e.*logistic regression classifier-based CINDEL [11]. Note that, because CINDEL is a classification model rather than a regression model, we used binary labels in this case, such that we assigned 1 to the top 20th percentile and 0 to the rest.

For featurization of nucleotide sequences, we used a previously described feature extraction procedure [9, 11], which included position-independent nucleotides and dinucleotides, position-dependent nucleotides and dinucleotides, melting temperature, GC counts, and the minimum self-folding free energy. We performed nested CV for model selection among the regularization parameter and hyperparameter configurations, the number of which is comparable

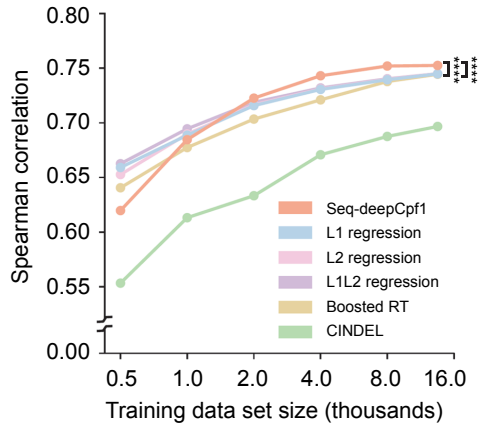

Figure 3.3: Nested CV of Cpf1 activity prediction models trained on different sizes of datasets. The Spearman correlation coefficients between experimentally obtained indel frequencies and predicted scores from Seq-deepCpf1 and other conventional machine learning approaches are plotted. For the sake of clarity, results from statistical significance testing are shown only between the best model and the two next-best models (Seq-deepCpf1 vs. L1L2 regression, ****P = 6.5 x 10-6; Seq-deepCpf1 vs. L2 regression, ****P = 5.5 × 106; Steiger's test).

to the number of hyperparameter configurations used for the development of Seq-deepCpf1 (180). For L1-, L2-, L1L2-regularized linear regression models and CINDEL, we searched over 250 points that were evenly spaced between 10-6 and 106 in log space to optimize the regularization parameter. For the Boosted RT, we searched over 225 models selected from the following hyperparameter configurations: the number of base estimators (chosen from [50, 100, 150, 200]), the maximum depth of the individual regression estimators (chosen from [2, 4, 6, 8, 10]), the minimum number of samples to split an internal node (chosen from [2, 4]), the minimum number of samples to be at a leaf node (chosen from [1, 2]), and the maximum number of features to consider when looking for the best split (chosen from [all features, the square root of all features, the binary logarithm of all features]).

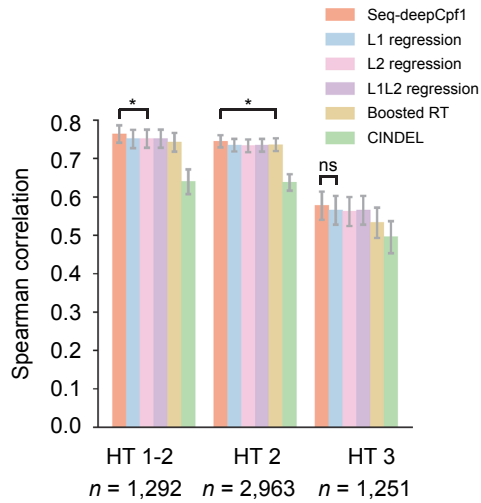

Figure 3.4: Performance comparison of prediction models. For three independent test datasets (HT 1-2, HT 2, HT 3), the Spearman correlation coefficients between measured indel frequencies and predicted Cpf1 activity scores are shown. For the sake of clarity, results from statistical significance testing are shown only for the pair of the best and the next-best models (left to right; *P = 0.015, *P = 0.026, and n.s. = not significant; Steiger's test).

3.2.3 Evaluation of Seq-deepCpf1

As the size of training data for the CV increased, the average Spearman correlation coefficients between experimentally obtained indel frequencies and predicted scores from Seq-deepCpf1 steadily increased up to 0.75 (Figure 3.3). Compared to conventional machine learning algorithms that performed best in the current SOTA approaches, the Spearman correlation of Seq-deepCpf1 in the CV was significantly higher than those of these conventional machine learning-based algorithms, especially when the training data size was sufficiently high (vs. L1L2 regression, P = 6.5 x 10-6; vs. L2 regression, P = 5.5 × 106). One of the reasons why CINDEL showed the worst performance is at least partly because CINDEL is a classification model, which leads to the loss of detailed information during the modeling. Furthermore, when these algorithms were evaluated using three different test datasets of Cpf1 activity (HT 1-2, HT 2, HT 3) that were never used during the training, the Spear-

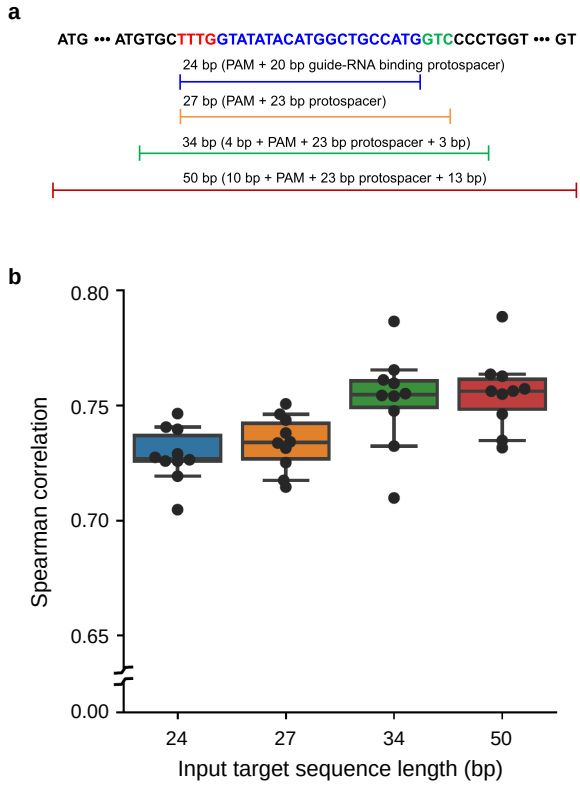

Figure 3.5: Seq-deepCpf1 models trained with different length of target sequences. (a) Candidate lengths for the input target sequence. (b) Nested CV results of Seq-deepCpf1 trained with different input target sequence lengths.

man correlations of Seq-deepCpf1 were significantly higher than those of the conventional machine learning-based algorithms (Figure 3.4). Taken together, these results suggest that deep learning outperforms these conventional machine learning methods for the prediction of Cpf1 activity based on target sequence composition.

For ablation studies, we conducted two additional experiments using the nested CV. First, we compared performance of Seq-deepCpf1 models trained with different length of target sequences 3.5. Target sequence lengths of 34-nucleotide and 50-nucleotide showed the best performance. Of the two, 34-nucleotide was chosen for the final input target sequence length because it led to a simpler model with fewer parameters, reducing the possibility of potential

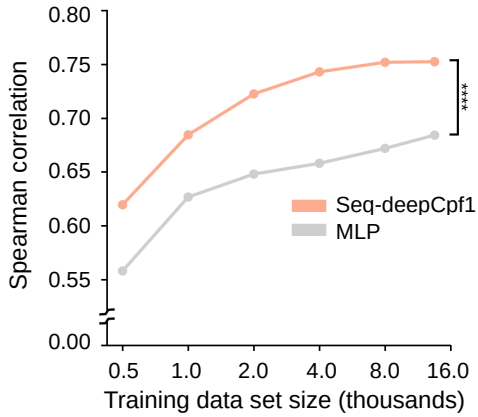

Figure 3.6: Nested CV results of Seq-deepCpf1 and a multi-layer perceptron (MLP; ****P = 4.1 x 108; Steiger's test).

overfitting. Second, we compared the performance of Seq-deepCpf1 and multi-layer perceptron, which lacks a convolution layer 3.6. The results showed that the convolution layer that can discover locally correlated patterns is crucial for predicting CRISPR activity. Note that We have also performed nested CV to select among 209 multi-layer perceptron models with similar architectures and number of parameters as Seq-deepCpf1.

3.2.4 Evaluation of DeepCpf1

When evaluated using datasets HEK-plasmid and HCT-plasmid as test datasets, DeepCpf1 showed substantially improved performance compared to other models (Figure 3.7). The DeepCpf1 prediction scores and measured indel frequency ranks showed high Spearman correlations both in HEK-plasmid and HCT-plasmid datasets, reaching 0.87 and 0.77, respectively (Figure 3.8). To the best of our knowledge they are the highest prediction performance achieved for any targeted nuclease. The results showed that DeepCpf1 scores can also be used to classify efficient endogenous targets into three groups (Figure 3.9). Based on two-sample Kolmogorov-Smirnov tests, indel frequencies were significantly different between the three groups for both HEK-plasmid (**P = 2.6

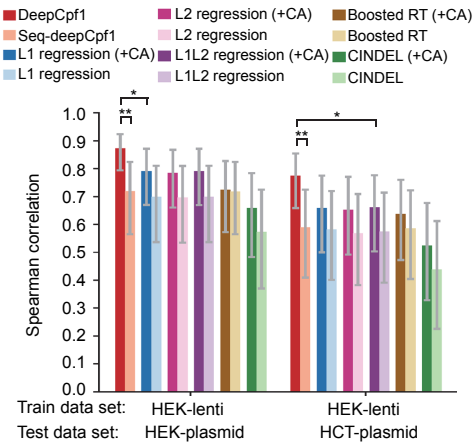

Figure 3.7: Performance comparison of DeepCpf1 with other prediction models in HEK293T cells (left, n=55) and HCT116 cells (right, n=66). The bar graph shows Spearman correlations between measured indel frequencies and predicted activity scores. For the sake of clarity, results from statistical significance testing are shown only for the DeepCpf1 versus Seq-deepCpf1 and DeepCpf1 versus next-best models (left to right; *P = 0.041, **P = 0.003, *P = 0.031, and **P = 0.005; Steiger's test).

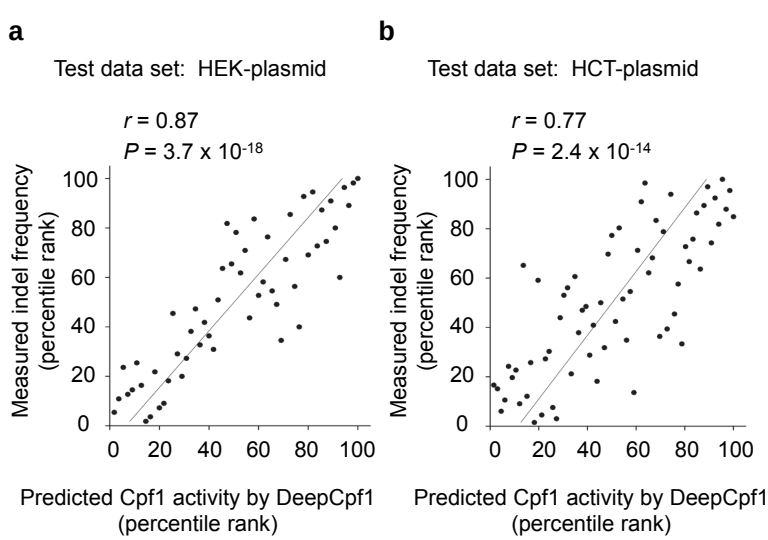

Figure 3.8: Correlation between DeepCpf1 prediction scores and measured indel frequency ranks at endogenous target sites in (a) HEK293T cells (n=55) and (b) HCT116 cells (n=66). The Spearman correlations (r) and P values (P) obtained using the student's t-test with n-2 degrees of freedom are shown.

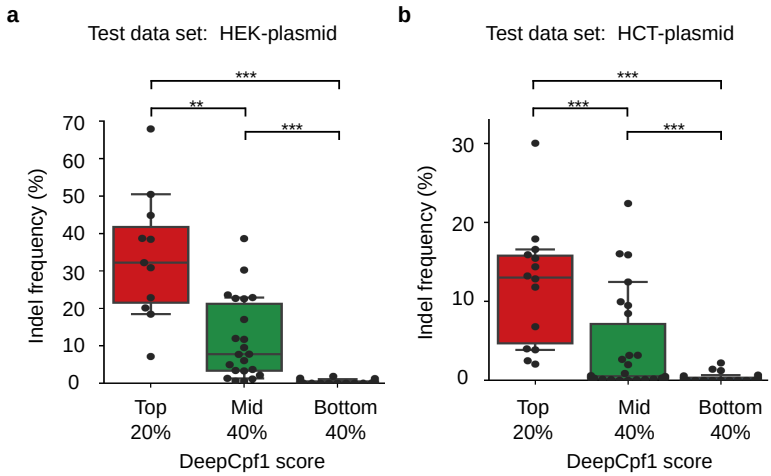

Figure 3.9: Classification of endogenous targets according to the DeepCpf1 scores (Top 20%, middle 40%, and bottom 40%).

x 10-3, Top 20% versus Mid 40%; ***P = 1.4 x 10-7, Top 20% versus Bottom 40%; ***P = 2.0 x 10-7, Mid 40% versus Bottom 40%) and for HCT-plasmid (***P = 9.8 x 10-4, Top 20% versus Mid 40%; ***P = 1.6 x 10-8, Top 20% versus Bottom 40%; ***P = 5.6 x 10-4, Mid 40% versus Bottom 40%).

We evaluated DeepCpf1 with independent endogenous datasets of different studies from independent laboratories [71, 72, 73]. The results showed Spearman correlations of 0.61, 0.70, and 0.79, suggesting excellent generalization performance (Figure 3.10). Furthermore, we also fine-tuned Seq-deepCpf1 with different training datasets *i.e.*, HEK-plasmid or HCT-plasmid, developing DeepCpf1-HEK-plasmid and DeepCpf1-HCT-plasmid, respectively. Both models showed high Spearman correlations ranging between 0.60 – 0.83 (Figure 3.10). It confirmed that the fine-tuning strategy of DeepCpf1 is a reliable and effective approach for improving CRISPR activity prediction.

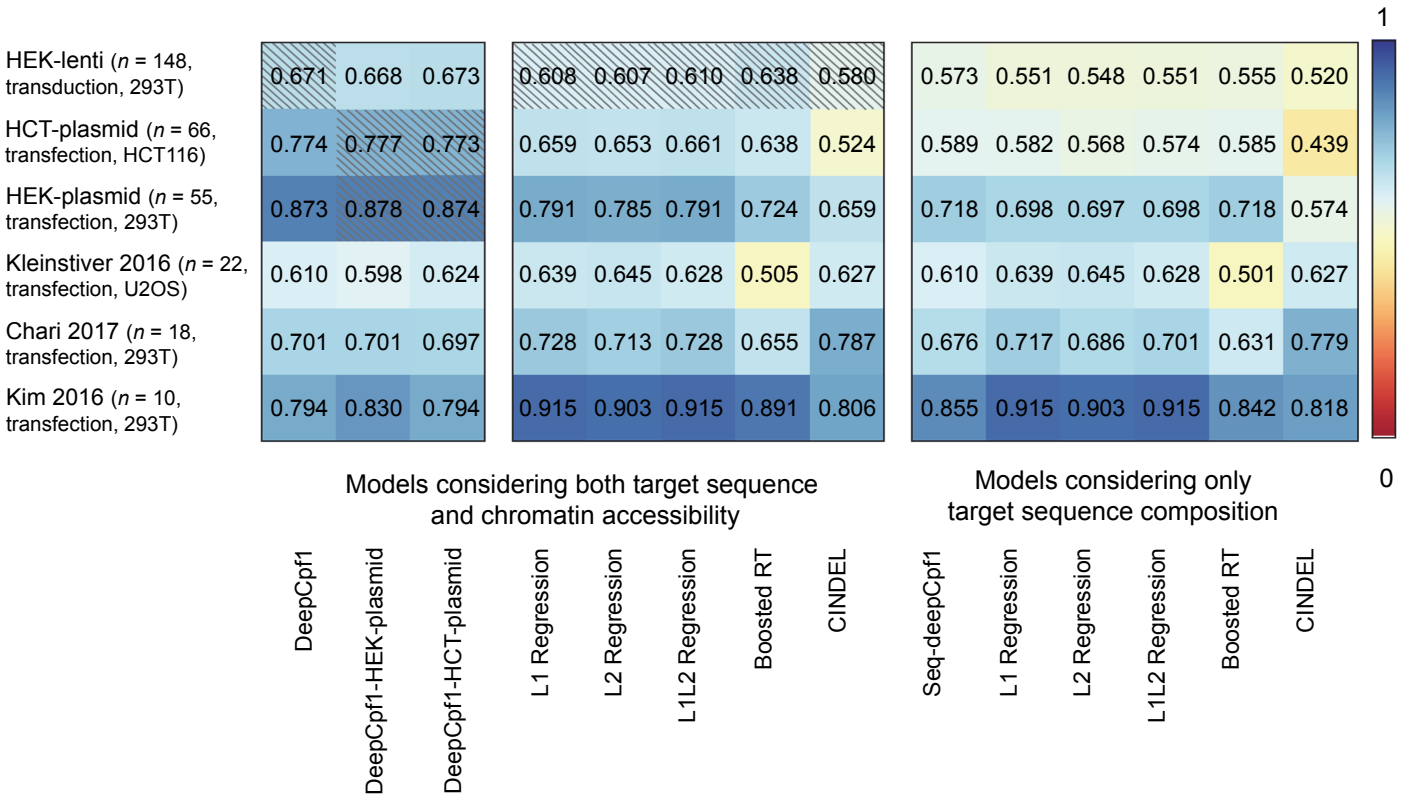

Figure 3.10: Heat map of Spearman correlations using different models and datasets. The test datasets are arranged vertically, whereas the prediction models are placed horizontally. Each cross-hatched box represents Spearman correlations of a model evaluated against a test dataset that includes its own training dataset.

3.3 Summary

Over the last few years, genome editing using the CRISPR system has become a crucial tool in biology. This chapter proposed a new deep learning-based algorithm that predicts CRISPR-Cpf1 activity at endogenous target sites based on both the sequence composition and chromatin accessibility of those sites. Incorporating a multi-modal CNN architecture, we found that deep learning outperforms shallow learning and that consideration of chromatin accessibility significantly improves prediction accuracy. When evaluated using endogenous test datasets *i.e.*HEK-plasmid and HCT-plasmid, the proposed algorithm showed substantially improved performance than other models, reaching Spearman correlations of 0.87 and 0.77, respectively. Furthermore, DeepCpf1 showed excellent generalization performance in other published datasets of different studies from independent laboratories. This high level of performance is in contrast to that of previously reported Cas9 activity prediction algorithms, which showed Spearman correlations of only 0.34 – 0.44 when independent test datasets of different studies from independent laboratories were used [68].

We have achieved a breakthrough in the development of CRISPR therapeutics for human diseases. To the best of our knowledge, this is the first work to propose a deep learning-based model for CRISPR-Cpf1 activity. Through the newly developed computational models, researchers can now more easily identify which of the thousands of sites within a gene should be targeted to achieve maximum efficiency. We believe this would remarkably reduce the amount of time, effort, and money invested in CRISPR genome editing, thereby brightening the future of precision medicine.

Chapter 4

Functional microRNA Target Prediction

Gene expression regulation is a key component of biological processes. The expression levels of different genes are controlled through several mechanisms. MicroRNAs (miRNAs) play a pivotal role in the post-transcriptional regulation of $\geq 60\%$ of human protein-coding genes [13]. MiRNAs are small non-coding RNAs that can bind to the target sites of messenger RNAs (mRNAs). This binding leads to the repression of efficient translation of mRNAs, thereby down-regulating the expression of target genes [74]. The effectiveness of each target site can vary depending on the site context and the binding stability [75]. While identifying functional targets of miRNAs is of utmost importance, their computational prediction remains a great challenge [76].

A miRNA can target multiple mRNAs by functioning as a sequence-specific guide. The binding is primarily directed through the interaction between the 5' ends of a miRNA, referred to as the "seed region," and the complementary 3' untranslated region (UTRs) of a target mRNA. Previous large-scale transcriptome studies have identified several target canonical site types that form Watson-Crick (WC) parings with the miRNA seed region [77]. The canonical site types include 6-mer sites (matching miRNA nucleotides 2-7), 7-mer-m8

sites (matching miRNA nucleotides 2-8), 7-mer-A1 sites (matching miRNA nucleotides 2-7 with an `A` opposite nucleotide 1), and 8-mer sites (matching miRNA nucleotides 2-8 with an `A` opposite nucleotide 1). More recent studies have also revealed that target non-canonical site types with `G:U` wobble pairings or gaps are also prevalent [76, 78].

A variety of computational algorithms have been proposed for functional miRNA target prediction [14]. Most of them follow a similar pipeline consisting of three stages (Figure 4.1). The first stage is the selection of candidate target sites (CTSs). Then, in the second stage, a prediction model identifies whether each miRNA-CTS pair is functional or non-functional. Finally, in the third stage, the predictions are post-processed to obtain a final prediction for the miRNA-mRNA pair. In general, a miRNA-mRNA pair is predicted to be functional if there is at least one miRNA-CTS pair predicted as functional.

Since a miRNA partially forms WC pairings to its cognate target mRNAs, it is vital to search for CTSs based on their binding characteristics in order to reduce the search space of a prediction algorithm. Given a miRNA-mRNA pair, computational algorithms use a sliding window to identify CTSs from 3' UTRs of the mRNA fulfilling specific criteria as follows: [15] considered (a) 7-mer of a mRNA that forms complete WC parings to a miRNA starting at nucleotide 1 or 2 and (b) a region that contains at least seven WC parings to a miRNA starting at nucleotide 1. [79] considered (a) 6-mer of a mRNA that forms complete WC parings to a miRNA starting at nucleotide 2, (b) 4-mer of a mRNA that forms three consecutive complete WC parings to a miRNA starting at nucleotide 13, and (c) 12-mer of a mRNA that forms eleven consecutive complete WC parings to a miRNA starting at nucleotide 4. [80] considered (a) a region that contains at least six WC parings to a miRNA at the nucleotides 1–10, (b) a region which contains at least seven WC parings to a miRNA at nucleotides 1–10, and (c) a region containing at least seven WC parings to a miRNA at nucleotides 2–10. Although these criteria

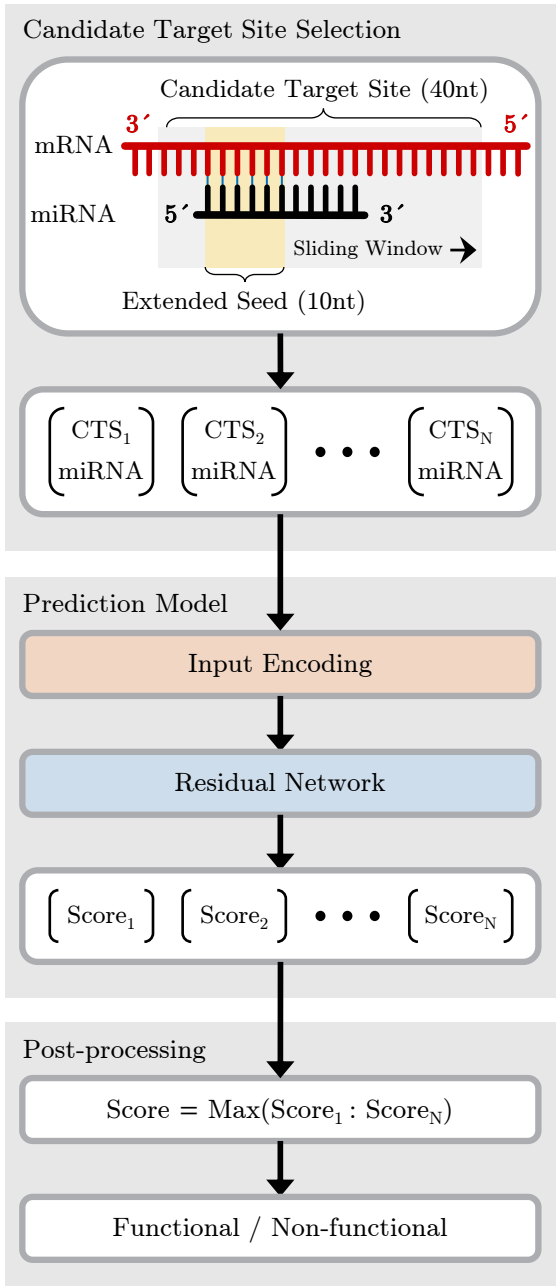

Figure 4.1: Schematic of functional miRNA target prediction algorithms.

enable a target prediction algorithm to reduce false positives by pre-processing CTSs based on empirical observations, they cannot capture non-canonical site patterns [76].

We can categorize existing miRNA target prediction models into two types: feature extraction-based and deep learning-based models. PITA [15], mirSVR [81], miRDB [82], and TargetScan [79] are feature extraction-based models which utilize different features individually. PITA utilizes site accessibility to compute a dynamic programming-based score. mirSVR utilizes sequence and contextual features to train a regression model. miRDB utilizes seed conservation features to train a SVM model. TargetScan utilizes seed conservation and structural features to train a regression model. Each feature engineering procedure depends on the research design; hence, it is difficult to define a consistent strategy. deepTarget [51] and miRAW [80] are deep learning-based models which utilize raw sequences as inputs. deepTarget utilizes RNN-based autoencoders to learn features; however, it considers only canonical site patterns and utilizes simulated training data to compensate for the number of negative pairs. miRAW utilizes multi-layer perceptron networks to learn features; however, it requires additional information, including binding and site accessibility energies. Although both models exploit CTSs to reduce the search space of algorithms, they ignore the information underlying CTSs, such as how each CTS forms pairings, mismatches, or bulges. To fully utilize the information underlying CTSs, we proposed a novel encoding scheme for miRNA-mRNA pairs.

While previous computational algorithms differ in CTS selection criteria and prediction models, they share certain significant limitations. They generally use conservative CTS selection criteria, which mainly focus on canonical site types. Because these conservative criteria only allow a limited number of non-canonical site types with few irregularities, they cannot capture the complete picture of functional miRNA target prediction [15]. In addition, the

majority of prediction models are based on feature extraction followed by the application of conventional machine learning classifiers (*e.g.*, linear regression and SVMs). They rely on the discovery of new hand-crafted features and often exploit additional information such as site location, accessibility, or minimum free energy [83]. Nevertheless, manual feature extraction requires laborious and time-consuming processes. This inevitably impedes the improvement of prediction models in terms of both efficiency and performance [7]. Several studies have recently proposed deep learning-based prediction models to automatically learn effective features [51, 80]. However, they still have not fully capitalized on information underlying miRNA-CTS interactions. Even though the CTS selection stage provides information on how each CTS forms pairings, mismatches, or gaps to bind with the miRNA seed region, previous studies only used miRNA-CTS sequences for their prediction models. This leaves considerable room for improvement and the development of a more effective data-driven computational algorithm.

In this section, we introduce TargetNet, a novel deep learning-based algorithm for functional miRNA target prediction. To address the previous limitations, TargetNet has three key components. First, it uses relaxed CTS selection criteria. Employing a sliding window, we align the extended seed region of a miRNA to the UTRs of a target mRNA. Then, we consider those aligned regions with at least 6 WC or wobble base pairings as the CTSs. Second, TargetNet uses a novel encoding scheme for miRNA-CTS sequences to incorporate the alignment information. This makes it easier for the DNN to learn features from the bindings formed by a miRNA-CTS pair. Third, TargetNet uses a deep residual network (ResNet) with one-dimensional convolutions as its prediction model [22]. Compared to previously used multi-layer perceptrons and RNNs, it is more effective for RNAs where local nucleotide motifs often have significant implications.

We used experimentally verified public datasets for empirical validation

[80, 75]. TargetNet was trained with miRNA-CTS pair datasets and evaluated using miRNA-mRNA pair datasets. Leveraged by these three key components, TargetNet demonstrates significant performance improvement in functional miRNA target classification over previous SOTA algorithms. Furthermore, top-ranked TargetNet prediction scores exhibit a high association with the level of miRNA-mRNA expression down-regulation, which demonstrates its great potential for distinguishing high-functional miRNA targets.

4.1 Methods

We propose TargetNet, a novel deep learning-based algorithm for functional microRNA target prediction (Algorithm 1). In the following, we will explain the details of its CTS selection, miRNA-CTS input encoding scheme, ResNet prediction model, and post-processing procedures.

Algorithm 1 TargetNet

Input: a miRNA-mRNA pair, S^{miRNA} and S^{mRNA}
Output: a prediction score, $0 \leq p^{\text{miRNA-mRNA}} \leq 1$

Stage 1: Candidate target site (CTS) selection (Section 4.1.1)

Use a 40-nt sliding window to obtain potential CTSs from the mRNA
▷ $S_i^{\text{CTS}} = \langle S_i^{\text{CTS-DS}}, S_i^{\text{CTS-ES}}, S_i^{\text{CTS-US}} \rangle$
Conduct a sequence alignment of miRNA-CTS pairs' extended seed regions
▷ $S^{\text{miRNA}} = \langle S^{\text{miRNA-ES}}, S^{\text{miRNA-DS}} \rangle$
▷ $\widetilde{S}^{\text{miRNA-ES}}, \widetilde{S}_i^{\text{CTS-ES}} = \text{Align}(S^{\text{miRNA-ES}}, S_i^{\text{CTS-ES}})$
Filter out miRNA-CTS pairs with the alignment scores < 6
▷ S^{miRNA} and S_i^{CTS}

Stage 2-1: Input encoding (Section 4.1.2)

Replace the extended region sequences with their alignment results
▷ $\widetilde{S}^{\text{miRNA}}$ and $\widetilde{S}_i^{\text{CTS}}$
Encode the miRNA and CTS sequences into 5-dimensional one-hot vectors
▷ $\mathbf{E}^{\text{miRNA}}$ and $\mathbf{E}_i^{\text{CTS}}$
Concatenate the encoded miRNA-CTS vectors with zero-paddings
▷ $\mathbf{E}_i = \text{Concat}(\langle \mathbf{0}^5, \mathbf{E}^{\text{miRNA}}, \mathbf{0}^{45-L_{\text{m}}} \rangle, \langle \mathbf{E}_i^{\text{CTS}}, \mathbf{0}^{50-L_{\text{c}}} \rangle)$

Stage 2-2: Prediction model (Section 4.1.3)

Feed the encoded the miRNA-CTS input into the input stem
▷ $\mathbf{H}_{i,1} = \text{Stem}(\mathbf{E}_i)$
Feed the output from the stem into the two residual blocks
▷ $\mathbf{H}_{i,2} = \text{ResBlock}_1(\mathbf{H}_{i,1})$ and $\mathbf{H}_{i,3} = \text{ResBlock}_2(\mathbf{H}_{i,2})$
Compute the output score for the miRNA-CTS pairs
▷ $p_i^{\text{miRNA-CTS}} = \text{Dense}(\text{MaxPool}(\mathbf{H}_{i,3}))$

Stage 3: Post-processing (Section 4.1.4)

Compute the final output score for the miRNA-mRNA pair
▷ $p^{\text{miRNA-mRNA}} = \max(p_1^{\text{miRNA-CTS}}, \cdots, p_N^{\text{miRNA-CTS}})$

4.1.1 Candidate Target Site Selection

Given a miRNA-mRNA pair, TargetNet first identifies CTSs that have the potential to be binding sites. We utilized a sliding window to scan through 3' UTRs of the mRNA (Figure 4.1). Since nucleotides beyond the seed are also important for miRNA-CTS interaction [84], we set the sliding window length to 40 nucleotides and its step length as one nucleotide. For each step, the sliding window produces a potential miRNA-CTS pair checked against the CTS selection criteria. In the following, miRNA and CTS sequences are denoted as:

$$\begin{aligned} S^{\text{miRNA}} &= (s_1^{\text{miRNA}}, \cdots, s_{L_{\text{mi}}}^{\text{miRNA}}), \\ S_i^{\text{CTS}} &= (s_{i,1}^{\text{CTS}}, \cdots, s_{i,40}^{\text{CTS}}), \\ s_j^{\text{miRNA}}, s_{i,j}^{\text{CTS}} &\in \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}, \end{aligned} \tag{4.1}$$

where S^{miRNA} and S_i^{CTS} are in the 5'-to-3' and 3'-to-5' directions, respectively. We use subscript i to indicate that there can be multiple CTSs for a given miRNA-mRNA pair. S^{miRNA} has a variable length, L_{mi}, which is 22 nucleotides on average while S_i^{CTS} has a fixed length of 40 nucleotides.

TargetNet adopts relaxed CTS selection criteria similar to those used in miRAW. First, we divide the S^{miRNA} into sub-sequences as:

$$\begin{aligned} S^{\text{miRNA}} &= \langle S^{\text{miRNA-ES}}, S^{\text{miRNA-DS}} \rangle, \\ S^{\text{miRNA-ES}} &= (s_1^{\text{miRNA}}, \cdots, s_{10}^{\text{miRNA}}), \\ S^{\text{miRNA-DS}} &= (s_{11}^{\text{miRNA}}, \cdots, s_{L_{\text{mi}}}^{\text{miRNA}}), \end{aligned} \tag{4.2}$$

where $S^{\text{miRNA-ES}}$ and $S^{\text{miRNA-DS}}$ denote the extended seed region and downstream nucleotides of a miRNA sequence, respectively. Similarly, we divide

S^{CTS} into sub-sequences as:

$$\begin{aligned} S_i^{\text{CTS}} &= \langle S_i^{\text{CTS-DS}}, S_i^{\text{CTS-ES}}, S_i^{\text{CTS-US}} \rangle, \\ S_i^{\text{CTS-DS}} &= (s_{i,1}^{\text{CTS}}, \cdots, s_{i,5}^{\text{CTS}}), \\ S_i^{\text{CTS-ES}} &= (s_{i,6}^{\text{CTS}}, \cdots, s_{i,15}^{\text{CTS}}), \\ S_i^{\text{CTS-US}} &= (s_{i,16}^{\text{CTS}}, \cdots, s_{i,40}^{\text{CTS}}), \end{aligned} \tag{4.3}$$

where $S_i^{\text{CTS-DS}}$, $S_i^{\text{CTS-ES}}$, and $S_i^{\text{CTS-US}}$ denote the downstream, extended seed region, and upstream nucleotides of a CTS sequence, respectively. Note that since S_i^{CTS} is in a 3'-to-5' direction, the former sub-sequence is toward the 3' end, and hence, it is called $S_i^{\text{CTS-DS}}$.

Then, we conduct a sequence alignment of the extended seed regions:

$$\widetilde{S}^{\text{miRNA-ES}}, \widetilde{S}_i^{\text{CTS-ES}} = \text{Align}(S^{\text{miRNA-ES}}, S_i^{\text{CTS-ES}}). \tag{4.4}$$

We find their best global alignment using a Biopython pairwise2 package [85]. The scoring matrix for the alignment is defined to produce a score of 1 for WC and wobble pairings and a score of 0 for the other pairings and gaps. If there are multiple best alignments, we use the first one obtained from the package. The alignment results, $\widetilde{S}^{\text{miRNA-ES}}$ and $\widetilde{S}^{\text{CTS-ES}}$, are composed of $s \in \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}, \texttt{-}\}$ representing four nucleotides and a gap. The relaxed CTS selection criteria are met if the alignment score is at least 6. It makes minimal assumptions regarding miRNA-CTS interactions; hence, it can accommodate a wide range of non-canonical sites and canonical sites.

4.1.2 Input Encoding

The most distinguishing component of TargetNet, which separates it from other deep learning-based methods, is the way it encodes a miRNA-CTS pair. Once the CTS selection is completed, the previous works use one-hot encoding to convert only sequences into numerical representations. In contrast, we pro-

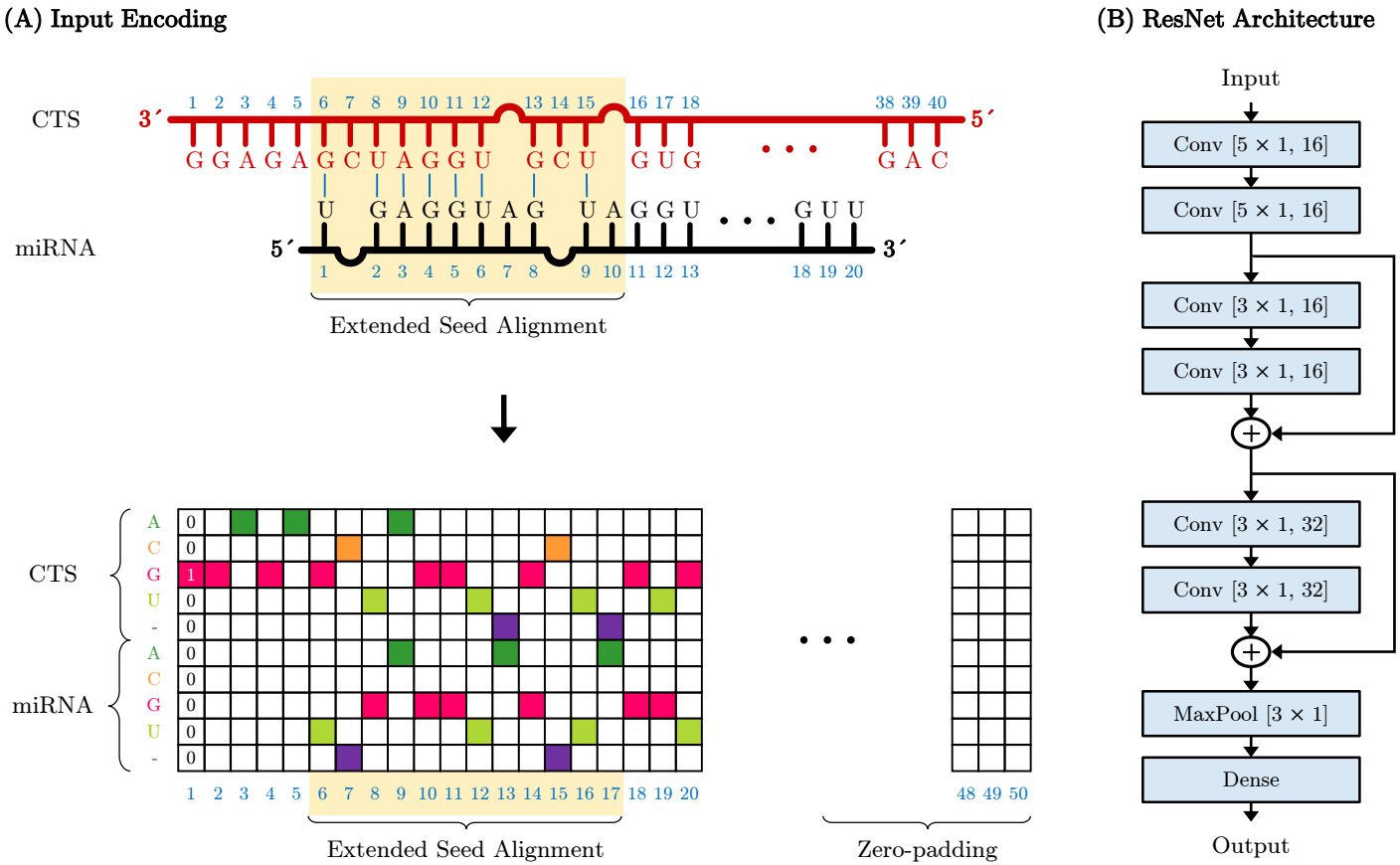

Figure 4.2: Overview of TargetNet prediction model. (A) Input encoding. (B) ResNet architecture.

pose a novel encoding scheme to incorporate additional information on how the extended seed regions of a miRNA-CTS pair are aligned and form bindings (Figure 4.2(A)).

TargetNet input encoding takes the alignment results of the extended seed regions (*i.e.*, $\widetilde{S}^{\text{miRNA-ES}}$ and $\widetilde{S}_i^{\text{CTS-ES}}$) in addition to the miRNA-CTS sequences (*i.e.*, S^{miRNA} and S_i^{CTS}). Specifically, we replace the extended region sequences, $S^{\text{miRNA-ES}}$ and $S_i^{\text{CTS-ES}}$, with their alignment results, $\widetilde{S}^{\text{miRNA-ES}}$ and $\widetilde{S}_i^{\text{CTS-ES}}$, and convert them using one-hot encoding:

$$\begin{aligned}
\mathbf{E}^{\text{miRNA}} &= \text{Encode}(\langle \widetilde{S}^{\text{miRNA-ES}}, S^{\text{miRNA-DS}} \rangle) \\
&= \langle \mathbf{e}_1^{\text{miRNA}}, \cdots, \mathbf{e}_{L_\text{m}}^{\text{miRNA}} \rangle, \\
\mathbf{E}_i^{\text{CTS}} &= \text{Encode}(\langle S_i^{\text{CTS-DS}}, \widetilde{S}_i^{\text{CTS-ES}}, S_i^{\text{CTS-US}} \rangle) \\
&= \langle \mathbf{e}_{i,1}^{\text{CTS}}, \cdots, \mathbf{e}_{i,L_\text{c}}^{\text{CTS}} \rangle,
\end{aligned} \tag{4.5}$$

where $\mathbf{e}_j^{\text{miRNA}}$ and $\mathbf{e}_{i,j}^{\text{CTS}}$ are 5-dimensional one-hot vectors indicating that the position is one of the 5 possible characters, $\{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}, \texttt{-}\}$. Both $\mathbf{E}^{\text{miRNA}}$ and $\mathbf{E}_i^{\text{CTS}}$ have variable lengths (*i.e.*, L_m and L_c) due to possible gaps in the alignment results.

Finally, we perform position-wise concatenation of $\mathbf{E}^{\text{miRNA}}$ and $\mathbf{E}_i^{\text{CTS}}$ with additional zero-padding $\mathbf{0}^\text{k} \in \mathbb{R}^{5\times\text{k}}$ as:

$$\begin{gathered}
\mathbf{E}_i = \text{Concat}(\hat{\mathbf{E}}^{\text{miRNA}}, \hat{\mathbf{E}}_i^{\text{CTS}}), \\
\hat{\mathbf{E}}^{\text{miRNA}} = \langle \mathbf{0}^5, \mathbf{E}^{\text{miRNA}}, \mathbf{0}^{45-L_\text{m}} \rangle, \quad \hat{\mathbf{E}}_i^{\text{CTS}} = \langle \mathbf{E}_i^{\text{CTS}}, \mathbf{0}^{50-L_\text{c}} \rangle.
\end{gathered} \tag{4.6}$$

Zero-paddings are used (1) to align the positions of the extended seed regions and (2) to make $\mathbf{E}_i$ be a 10-by-50 sized vector. The advantage of the proposed input encoding is that it makes it more accessible for the following ResNet to fully capitalize on information underlying miRNA-CTS interactions. The input vector can now represent the miRNA-CTS sequences and how pairings, mismatches, or gaps are formed within their extended seed regions.

4.1.3 Residual Network

Motivated by the recent successes of ResNets in computer vision problems [22], we use a ResNet as a prediction model for TargetNet (Figure 4.2(B)). To the best of our knowledge, this is the first work to use the ResNet for functional miRNA target prediction.

Let $f_{k,n}$ be a one-dimensional convolution layer, where k and n denote the filter lengths and the number of filters, respectively. The filters are convolved along the length axis and learned to find motifs. Compared to other layers, it can be more effective for RNAs where motifs have significant implications. Furthermore, the filters can also be understood to be learnable position-weighted matrices used in conventional techniques [7]. Each convolution layer is followed by a rectified linear unit activation function and dropout regularization [37]. We use zero-padding to keep the output sizes unchanged.

First, our model has an input stem (denoted as Stem) which takes the encoded miRNA-CTS vector as input:

$$\mathbf{H}_{i,1} = \text{Stem}(\mathbf{E}_i) = f_{5,16}(f_{5,16}(\mathbf{E}_i, \mathbf{W}_1), \mathbf{W}_2), \tag{4.7}$$

Then, its output is feed into the two residual blocks (denoted as ResBlock_1 and ResBlock_2), each consisting of two convolution layers:

$$\begin{aligned} \mathbf{H}_{i,2} &= \text{ResBlock}_1(\mathbf{H}_{i,1}) = \mathbf{H}_{i,1} + f_{3,16}(f_{3,16}(\mathbf{H}_{i,1}, \mathbf{W}_3), \mathbf{W}_4), \\ \mathbf{H}_{i,3} &= \text{ResBlock}_2(\mathbf{H}_{i,2}) = \mathbf{H}_{i,2} + f_{3,32}(f_{3,32}(\mathbf{H}_{i,2}, \mathbf{W}_5), \mathbf{W}_6), \end{aligned} \tag{4.8}$$

where $\mathbf{W}_l$ is the learnable parameters of the l-th layer. Let $\mathcal{F}(\mathbf{X})$ be an optimal function to be learned by a group of layers. While standard layers (*e.g.*, input stem) are formulated to learn $\mathcal{F}(\mathbf{X})$ directly, residual blocks are reformulated with skip connections to learn its residual function, $\mathcal{R}(\mathbf{X}) := \mathcal{F}(\mathbf{X}) - \mathbf{X}$. It has been shown to ease learning by enabling us (1) to pre-condition $\mathcal{F}(\mathbf{X})$ to be closer to an identity mapping and (2) to directly propagate forward and

backward signals [86].

Finally, we compute the output score $0 \leq p_i^{\text{miRNA-CTS}} \leq 1$, which indicates how likely a given miRNA-CTS pair is functional. $\mathbf{H}_{i,3}$ is fed into a max-pooling layer and a dense layer (denoted as MaxPool and Dense, respectively) as:

$$p_i^{\text{miRNA-CTS}} = \text{Dense}(\text{MaxPool}(\mathbf{H}_{i,3}), \mathbf{W}_7), \tag{4.9}$$

where $\mathbf{W}_7$ denotes the learnable parameters of the dense layer. The max-pooling layer reduces the output sizes by computing the channel-wise maximum value for non-overlapping windows of size 3. We use a sigmoid function as an activation function for the dense layer.

For training of the ResNet prediction model, we use binary cross-entropy objective function defined as:

$$\mathcal{L} = -(y \log(p^{\text{miRNA-CTS}}) + (1 - y) \log(1 - p^{\text{miRNA-CTS}})), \tag{4.10}$$

where $y \in \{0, 1\}$ specifies the label for a given miRNA-CTS pair. Note that we use miRNA-CTS pair datasets to train the prediction model rather than miRNA-mRNA pair datasets (Section 4.2.1). We use Adam optimizer [33], a training epoch size of 50, a mini-batch size of 256, a learning rate of 0.001, and a dropout probability of 0.5.

4.1.4 Post-processing

In the final stage, the output scores are post-processed to obtain a final score for a miRNA-mRNA pair. We use the maximum value from the scores for each miRNA-CTS pair. Formally, if there are N CTSs in a mRNA, we get output scores $p_1^{\text{miRNA-CTS}}, \cdots, p_N^{\text{miRNA-CTS}}$ for each CTS. The final score $p^{\text{miRNA-mRNA}}$ for a miRNA-mRNA pair is reported by:

$$p^{\text{miRNA-mRNA}} = \max(p_1^{\text{miRNA-CTS}}, \cdots, p_N^{\text{miRNA-CTS}}). \tag{4.11}$$

This results in the prediction of a miRNA-mRNA pair as functional if there is at least one functional miRNA-CTS pair. For the binary classification of functional targets, we used a threshold of 0.5 to binarize the final score $p^{\text{miRNA-mRNA}}$. Note that in contrast to miRAW, we do not exploit any additional filters based on site accessibility or minimum free energy.

4.2 Experiment Results

In the following subsections, we will explain the experiment datasets, evaluation results for both binary functional target classification and real-valued high-functional target regression, and ablation studies for different components of TargetNet. All models were implemented using PyTorch library [87].

4.2.1 Datasets

miRNA-mRNA Pair Datasets

The complete TargetNet algorithm was evaluated with two types of experimentally verified miRNA-mRNA pair datasets, (1) miRAW and (2) log fold change (LFC) test datasets.

First, we used miRAW test datasets with binary labels indicating functional and non-functional targets [80]. They originated from DIANA-TarBase [88] and MirTarBase [89] databases. After removing duplicated samples, they consisted of 309,912 positive and 1,096 negative miRNA-mRNA pairs. Then, they were split in half and used for the train-validation (Section 4.2.1) and test datasets, respectively. The authors generated ten randomly sampled test datasets, consisting of 548 positive and 548 negative pairs. The miRAW test datasets can help us evaluate the functional miRNA target classification performance of TargetNet.

Second, we used LFC test datasets with real-valued labels indicating the level of functionality of miRNA targets [75]. They contained 32,499 miRNA-mRNA pairs from 11 microarrays. In each microarray, a miRNA was transfected into HeLa cells, and the log fold change of mRNA expression down-regulation was measured. Thus, more negative labels indicate more functional miRNA-mRNA pairs. The LFC test datasets can help us to evaluate how well TargetNet distinguishes high-functional miRNA targets.

miRNA-CTS Pair Datasets

The prediction model of TargetNet was trained with miRAW miRNA-CTS pair datasets. To obtain miRNA-CTS pairs from the excluded miRNA-mRNA pairs, the authors pre-processed the positive pairs in two ways. One was cross-referencing with binding sites from PAR-CLIP [90] and CLASH [91], and keeping miRNA-CTS pairs that form stable duplexes. The other was cross-referencing with conserved sites from TargetScanHuman [79]. The negative pairs were pre-processed using a sliding window to identify miRNA-CTS pairs forming stable duplexes. The stability was measured with RNACofold [92] by checking whether their secondary structures produce negative free energy.

Since the miRAW dataset split was based on miRNA-mRNA pairs (Section 4.2.1), similar miRNAs can be distributed in both train-validation and test datasets. In other words, there can be a concern that experiments with miRAW datasets only enable us to evaluate generalization performance in terms of different targets. Thus, we used independent LFC test datasets to further evaluate generalization performance in terms of other miRNAs. We filtered out miRNA-CTS pairs so that no two miRNAs from the miRAW train-validation and LFC test datasets have Levenshtein edit distance lower than 7. Then, we randomly selected 20 miRNAs and used 2,385 positive and 2,264 negative miRNA-CTS pairs as a validation set. The remaining pairs containing 26,803 positive and 27,341 negative pairs were used as a training set.

4.2.2 Classification of Functional and Non-functional Targets

We compared the performance of TargetNet with six SOTA target prediction algorithms. PITA [15], miRDB [82], miRanda [81], and TargetScan [79] are feature extraction-based algorithms. deepTarget [51] and miRAW [80] are deep learning-based algorithms. First, we compared their classification performance. We used six binary classification evaluation metrics, including F1 score, accuracy, precision, recall, specificity, and negative precision.

Table 4.1: Functional miRNA target classification results.

Method	F1 Score	Accuracy	Precision	Recall	Specificity
PITA	0.2162	0.5053	0.5196	0.1365	0.8741
miRDB	0.2110	0.5373	0.7135	0.1239	**0.9507**
miRanda	0.3568	0.5001	0.4997	0.2775	0.7226
TargetScan	0.4712	0.5577	0.5852	0.3945	0.7208
deepTarget	0.4904	0.6521	**0.8332**	0.3477	0.9354
miRAW	0.7289	0.7055	0.6749	0.7923	0.6186
TargetNet	**0.7739**	**0.7251**	0.6572	**0.9411**	0.5091

Table 4.1 presents the averaged classification performance on the miRAW test datasets. The results demonstrated that TargetNet outperforms the other SOTA algorithms in general performance measures, namely, F1 score and accuracy. The F1 score and accuracy differences between TargetNet and the second-best algorithm, miRAW, were statistically significant with p-values of 1.1×10^{-5} and 2.1×10^{-3}, respectively. For the evaluation of statistical significance, we used the two-sample Kolmogorov-Smirnov test [93]. Its rejected null hypothesis is that the two independent groups of samples (*e.g.*, F1 scores obtained from TargetNet and miRAW) are from the same distributions.

While the other models, PITA, miRDB, miRanda, TargetScan, and deepTarget, exhibited high specificity, they failed to classify a large number of functional miRNA targets correctly. This is mainly due to their conservative CTS selection criteria, which neglect the majority of non-canonical site types. On the other hand, TargetNet and miRAW used more relaxed CTS selection criteria, which resulted in their higher recall. Since TargetNet and miRAW share similar CTS selection criteria and the same training dataset, their comparison can illustrate the effectiveness of the prediction model. The performance improvement indicates that the proposed encoding scheme and ResNet architecture can better capture the information underlying miRNA-CTS interactions.

Table 4.2: Classification results with different CTS selection criteria.

Criteria	Method	F1 Score	Precision	Recall
miRAW-6-1:10	miRAW	0.7289	0.6749	0.7923
	TargetNet	0.7491	0.7277	0.6945
miRAW-7-1:10	miRAW	0.7069	0.7188	0.6956
	TargetNet	0.7388	0.7242	0.7014
miRAW-7-2:10	miRAW	0.7222	0.7193	0.7255
	TargetNet	0.7422	0.7282	0.7056
TargetScan	miRAW	0.5325	0.7859	0.4029
	TargetNet	0.6747	0.6923	0.7155
PITA	miRAW	0.5694	0.7654	0.4537
	TargetNet	0.6901	0.6979	0.7081

We investigated more closely how CTS selection criteria affect classification performance on miRAW test datasets. While keeping the other stages intact, we evaluated miRAW and TargetNet using five different CTS selection criteria. Note that miRAW-6-1:10 is identical to the one used in TargetNet, except that it uses a sliding window step length of 5 nucleotides. From Table 4.2, we can make the following observations. First, regardless of the CTS selection criteria, TargetNet consistently outperformed miRAW in terms of the F1 score. This once again demonstrated the effectiveness of the proposed model. Second, using more conservative criteria deteriorates the performance of both miRAW and TargetNet. It filters out most of the candidate targets before using the prediction models, thus resulting in significant drops of recall and F1 score.

4.2.3 Distinguishing High-functional Targets

Next, we examined the association between the level of expression down-regulation and the top-ranked prediction scores over a broad range of cutoffs. We used independent LFC test datasets that do not contain any miRNAs similar to those in the miRAW train-validation dataset. Note that miRDB and TargetScan were trained with target expression data, thus, have a significant

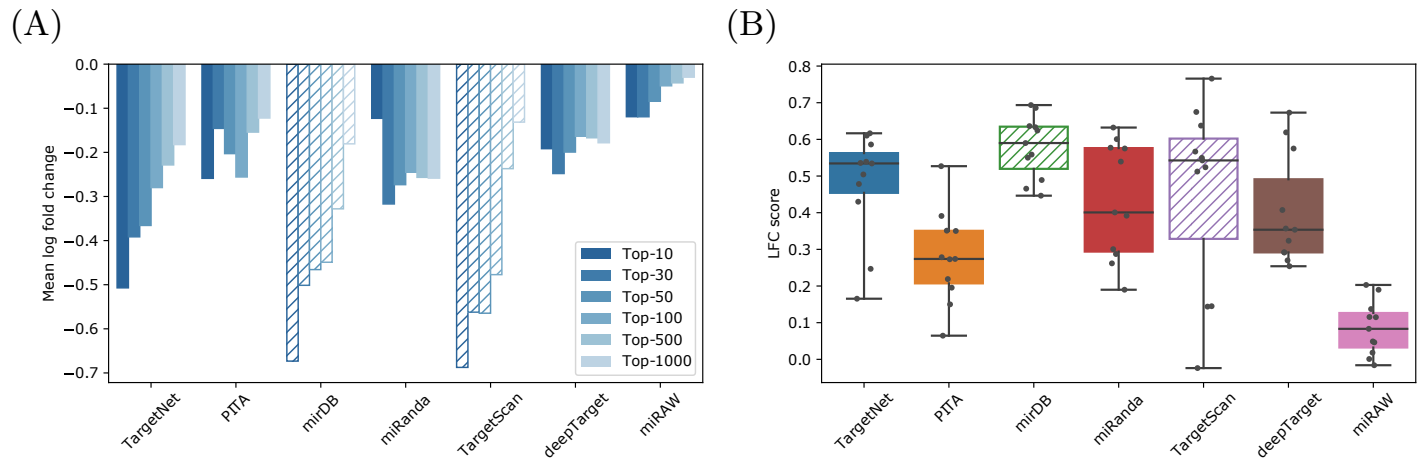

Figure 4.3: Performance of target prediction algorithms on distinguishing high-functional targets. (A) Mean log fold changes of miRNA-mRNA expression down-regulation for top-ranked predictions. (B) Per-miRNA LFC score distributions for top-ranked predictions. The hatched bars and boxes indicate the results from algorithms trained using target expression data.

advantage in this task. In Figure 4.3, we used hatched boxes to differentiate their results from the others.

Figure 4.3(A) presents the mean expression log fold changes of the top-ranked predictions. We ranked 32,499 miRNA-mRNA pairs from the LFC dataset according to the scores of each algorithm. Then, from the top-ranked predictions, the averages of their expression log fold change values were plotted. The results showed that the top TargetNet predictions were highly associated with the level of target expression down-regulation. As we select a smaller number of top-ranked predictions from TargetNet, we can observe more repressed, thus, more functional targets. Even though the proposed algorithm does not exploit any expression training data, it shows comparable performance with miRDB and TargetScan. Compared to the other algorithms, TargetNet significantly outperformed both feature extraction-based and deep learning-based algorithms.

To more quantitatively evaluate the distinguishing high-functional targets, we propose a novel LFC score defined as:

$$
\begin{aligned}
s_{\mathrm{LFC}} &= 1 - \sum_{k} \left(\frac{m_k^{\mathrm{true}} - m_k^{\mathrm{pred}}}{m_k^{\mathrm{true}}}\right)^2, \\
k &\in \{10, 30, 50, 100, 500, 1000\},
\end{aligned}
\tag{4.12}
$$

where m_k^{true} and m_k^{pred} denote the mean log fold change of the top-k predictions in terms of true expression values and prediction scores, respectively. The proposed score quantifies the strength of the association between the level of expression down-regulation and the top-ranked prediction scores. The better a prediction algorithm is, the more its scores resemble the true values and its s_{LFC} is closer to 1. Figure 4.3(B) presents the per-miRNA s_{LFC} distributions for top-ranked predictions. TargetNet provides competitive performance with miRDB and miRanda. The results demonstrated its great potential for distinguishing high-functional miRNA targets.

Table 4.3: Ablation studies on the prediction model of TargetNet.

	Alignment Encoding	Skip Connection	Number of Blocks	Widening Factor	F1 Score
BASE	**TRUE**	**TRUE**	**2**	**1**	**0.8230**
(A)	FALSE	TRUE	2	1	0.7204
(B)	TRUE	FALSE	2	1	0.7743
(C)	TRUE	TRUE	2	0.5	0.7955
	TRUE	TRUE	2	2	0.8102
	TRUE	TRUE	4	1	0.7362

4.2.4 Ablation Studies

Table 4.3 presents the results of the ablation studies to better understand TargetNet prediction models. We varied the components of the base model and measured the classification performance on the miRAW validation set.

In row (A), we can observe that disregarding the proposed alignment input encoding significantly degrades the model performance. This suggests that incorporating extended seed region alignments provides invaluable information for functional miRNA prediction. In row (B), we replaced our ResNet model with a conventional CNN by removing the skip connections. Note that the compared model has the same number of parameters as the base model. Thus, the performance drop confirms that the residual connection enables more efficient training of the model. Finally, in rows (C), we varied the number of blocks and the number of filters by a widening factor. While doubling the number of filters produced similar results to the base model, other model complexity alterations resulted in inferior classification performance.

4.3 Summary

We proposed a deep learning-based algorithm for functional miRNA target prediction. TargetNet adopts relaxed CTS selection criteria to accommodate a variety of non-canonical and canonical site types. We introduced a novel input encoding scheme to embrace both miRNA-CTS sequences and how their extended seed regions form bindings. Then, we used ResNet to capture the information underlying miRNA-CTS interactions. Our experimental results supported that TargetNet demonstrates not only significant performance improvement in functional miRNA target classification but also its top-ranked prediction scores show a high association with the level of miRNA-mRNA expression down-regulation.

Chapter 5

Self-supervised Learning of Protein Representations

Proteins consisting of amino acids are among the most versatile molecules in living organisms. They serve vital functions in biological mechanisms, *e.g.*, transporting other molecules and providing immune protection [94]. The versatility of proteins is generally attributed to their diverse structures. Proteins naturally fold into three-dimensional structures determined by their amino acid sequences. These structures have a direct impact on their functions.

With the development of next-generation sequencing technologies, protein sequences have become relatively more accessible. However, annotating a sequence with meaningful attributes is still time-consuming and resource-intensive. To bridge the exponentially growing gap between the numbers of unlabeled and labeled protein sequences, various *in silico* approaches have been widely adopted for predicting the characteristics of proteins [16].

Sequence alignment is a key technique in computational protein biology. Alignment-based methods are used to compare protein sequences using carefully designed scoring matrices or HMMs [95, 96]. Correct alignments can group similar sequences, provide information on conserved regions, and help investigate uncharacterized proteins. However, its computational complexity

increases exponentially with the number of proteins, and it has difficulties in identifying distantly related proteins. Homologous proteins sharing a common evolutionary ancestor can have high sequence-level variations [97]. Therefore, a simple comparison of sequence similarities often fails to capture the global structural and functional similarities of proteins.

Building upon the success of deep learning, several studies proposed deep learning algorithms for computational protein biology. Some of these algorithms only use raw protein sequences, whereas others may use additional features [7]. They have advanced the SOTA for various protein biology tasks. However, development of these algorithms requires highly task-specific processes, *e.g.*, training a randomly initialized model from scratch. It demands careful consideration of the model architectures and hyperparameters tailored for each task. Additional features, such as alignment-based features or known structural traits, may also be required for some tasks [19].

Transfer learning is an important cornerstone of deep learning. For example, in natural language processing, word representations are pre-trained using a huge amount of unlabeled text [98, 99]. The learned information can be transferred to a wide range of tasks by training task-specific models on top of the pre-trained word representations. The crux of transfer learning is how to pre-train representations. Several studies have proposed language model (LM)-based approaches that can exploit unlabeled data. The key idea is that ideal representations must convey syntactic and semantic information, and thus we must be able to use a representation of a token to predict its neighboring tokens.

For example, embeddings from language models (ELMo) learned contextualized representations by adopting forward and reverse RNNs [100]. Given a sequence of tokens without additional labels, the forward RNN sequentially processes the sequence left-to-right. It is trained to predict the next token, given its history. The reverse RNN is similar but processes the sequence in

reverse, right-to-left. After the pre-training, the hidden states of both RNNs are merged into a single vector representation for each token. Thus, the same token can be transformed into different representations based on its context. The major limitation of ELMo is that RNNs are trained using unidirectional LM and simply combined afterward. As valuable information often comes from both directions, unidirectional LM is inevitably suboptimal. To address this problem, BERT was proposed to pre-train bidirectional natural language representations using the TFM model [99]. Instead of the conventional LM, BERT utilizes an masked language modeling (MLM) pre-training task. It masks some input tokens at random and trains the model to predict them from the context. In addition, BERT includes a complementary NLP-specific pre-training task, next sentence prediction, which enables the learning of sentence relationships by training a model to predict whether a given pair of sentences is consecutive.

Now, the natural question is: can protein biology also take advantage of semi-supervised learning? According to the linguistic hypothesis [57], naturally occurring proteins are not purely random. Evolutionary pressure constrains them to a learnable manifold where indispensable structures and functions are maintained. Thus, by observing many unlabeled protein sequences, we can obtain an implicit understanding of the language of proteins. Several studies have recently proposed LM-based pre-training methods for protein transfer learning [18, 101, 102, 103]. Taking advantage of a large number of unlabeled protein sequences, it was demonstrated that pre-trained protein representations convey biochemical, structural, and evolutionary information. Therefore, according to the recent benchmark results from tasks assessing protein embeddings (TAPE), pre-trained representations can help improve model performance in various protein biology tasks [19].

The most closely related previous methods to our work are P-ELMo [17] and UniRep [18]. They have some common limitations and still often lag behind task-specific models [19]. First, they learn unidirectional representations

from unlabeled datasets. Unidirectional representations are sub-optimal for numerous protein biology tasks, where it is crucial to assimilate global information from both directions. Note that we do not consider combination of two unidirectional representations as bidirectional representations since they were simply combined after the unidirectional pre-training. Second, most pre-training methods solely rely on LM to learn from unlabeled protein sequences. Although LM is a simple and effective task, a complementary pre-training task tailored for each data modality has been often the key to further improve the quality of representations in other domains. For instance, in NLP, BERT adopted the next sentence prediction task. In another example, ALBERT devised a complementary sentence order prediction task to model the inter-sentence coherence and yielded consistent performance improvements for downstream tasks [104]. Similarly, a complementary protein-specific task for pre-training might be necessary to better capture the information contained within unlabeled proteins.

In this section, we introduce a novel pre-training method for protein sequence modeling and name it **PLUS**, which stands for **P**rotein sequence representations **L**earned **U**sing **S**tructural information. PLUS consists of masked language modeling (MLM) and an additional complementary protein-specific pre-training task, same-family prediction (SFP). SFP leverages computationally clustered protein families [105] and helps to better capture the global structural information within unlabeled protein sequences. We use PLUS to pre-train a BiRNN and refer to the resulting model as PLUS-RNN. This pre-trained universal model is fine-tuned on various downstream tasks without training randomly initialized task-specific models from scratch. Our experiment results demonstrate that PLUS-RNN outperforms other models of similar size solely pre-trained with the conventional LM in six out of seven widely used protein biology tasks. The seven tasks include three protein-level classification, two protein-level regression, and two amino-acid-level classification.

PLUS provides a novel way to exploit evolutionary relationships among unlabeled proteins and is broadly applicable across protein biology tasks. Furthermore, we also compare PLUS with larger SOTA protein LMs using the additional heat shock protein (HSP) identification task. Finally, we also present the results from our qualitative interpretation analyses to illustrate the strengths of PLUS-RNN.

5.1 Methods

5.1.1 Pre-training Procedure

We introduce PLUS (Figure 5.1), a novel pre-training method for protein sequence modeling. PLUS can be used to pre-train various model architectures that transform a protein sequence $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ into a sequence of bidirectional representations $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_n]$. In this work, we use it to pre-train a BiRNN and refer to the resulting model as PLUS-RNN. PLUS consists of two pre-training tasks: MLM and SFP. It can help the model to learn structurally contextualized bidirectional representations. The complete pre-training objective loss is defined as:

$$\mathcal{L}_{\text{PT}} = \lambda_{\text{PT}} \mathcal{L}_{\text{MLM}} + (1 - \lambda_{\text{PT}}) \mathcal{L}_{\text{SFP}} \tag{5.1}$$

where $\mathcal{L}_{\text{MLM}}$ and $\mathcal{L}_{\text{SFP}}$ are the MLM and SFP objective losses, respectively. We use a hyperparameter λ_{PT} to control the relative importance of the MLM and SFP objective losses.

Task #1: Masked Language Modeling (MLM)

Given a protein sequence, X, we randomly select 15% of the amino acids. Then, for each selected amino acid $\mathbf{x}_i$, we randomly perform one of the following procedures. For 80% of the time, we replace $\mathbf{x}_i$ with the token denoting an unspecified amino acid. For 10% of the time, we randomly replace $\mathbf{x}_i$ with one of the 20 proteinogenic amino acids. Finally, for the remaining 10%, we keep $\mathbf{x}_i$ intact. This is to bias the learning toward the true amino acids. For the probabilities of masking actions, we follow those used in the pre-training of BERT [99].

PLUS-RNN transforms a masked protein sequence, $\hat{X}$, into a sequence of bidirectional representations. Then, we use an MLM decoder to compute log probabilities for $\widetilde{X}$ over 20 proteinogenic amino acid types. The MLM task

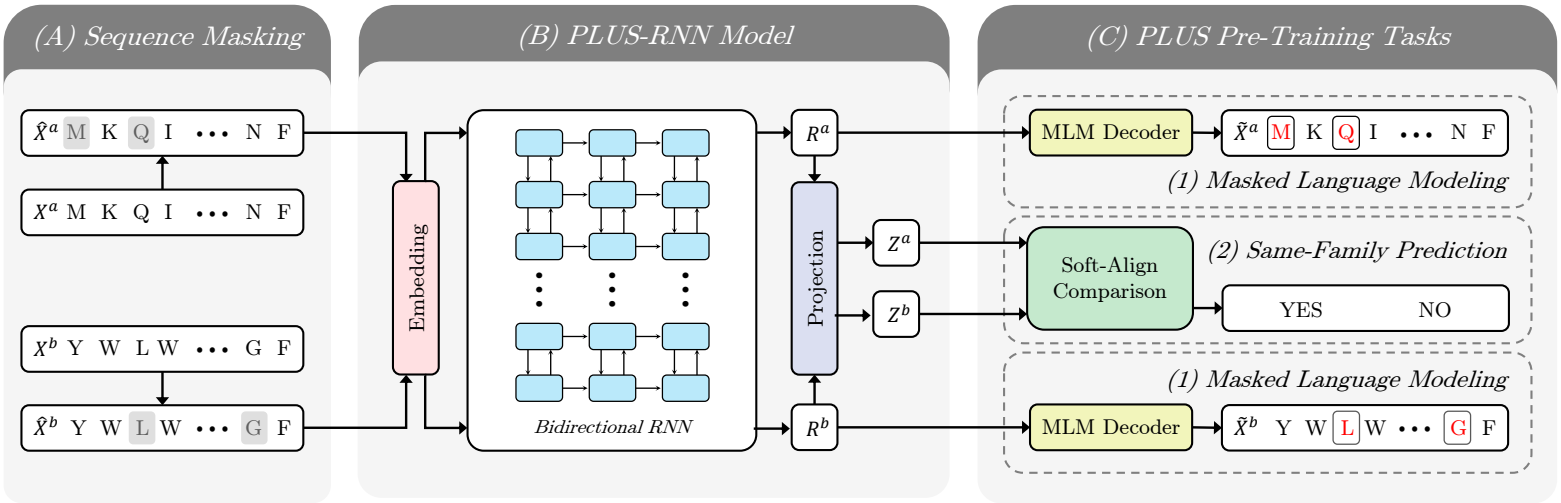

Figure 5.1: Overview of PLUS pre-training method. (A) We randomly mask 15% of amino acids (gray boxes) in each protein. (B) PLUS-RNN transforms protein sequences into sequences of bidirectional representations. (C) PLUS consists of two pre-training tasks. Masked language modeling trains a model to predict the masked amino acids (colored red within white boxes) given their contexts. Same-family prediction trains a model to predict whether a pair of proteins belongs to the same protein family.

trains the model to maximize the probabilities corresponding to the masked amino acids. As the model is designed to accurately predict randomly masked amino acids, the learned bidirectional representations must convey syntactic and semantic information within protein sequences.

Task #2: Same-Family Prediction (SFP)

Considering that additional pre-training tasks has been often the key for improving the quality of representations in other domains [99, 104], we devise a complementary protein-specific pre-training task. The SFP task trains a model to predict whether a given protein pair belongs to the same protein family. The protein family labels provide weak structural information and help the model learn structurally contextualized representations. Note that PLUS is still a semi-supervised learning method; it is supervised by computationally clustered weak labels rather than human-annotated labels.

We randomly sample two protein sequences, X^a and X^b, from the training dataset. In 50% of the cases, two sequences are sampled from the same protein family. For the other 50%, they are randomly sampled from different families. PLUS-RNN transforms the protein pair into sequences of representations $Z^a = [\mathbf{z}_1^a, \cdots, \mathbf{z}_{n_1}^a]$ and $Z^b = [\mathbf{z}_1^b, \cdots, \mathbf{z}_{n_2}^b]$. Then, we use a soft-align comparison [17] to compute their similarity score, $\hat{c}$, as a negative weighted sum of $l1$-distances between every $\mathbf{z}_i^a$ and $\mathbf{z}_j^b$ pair:

$$\hat{c} = -\frac{1}{C}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\omega_{ij}\left\|\mathbf{z}_i^a - \mathbf{z}_j^b\right\|_1, \quad C = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\omega_{ij}, \tag{5.2}$$

where weight ω_{ij} of each $l1$-distance is computed as

$$\begin{gathered}\omega_{ij} = 1 - (1-\alpha_{ij})(1-\beta_{ij}), \\ \alpha_{ij} = \frac{\exp(-\left\|\mathbf{z}_i^a - \mathbf{z}_j^b\right\|_1)}{\sum_{k=1}^{n_2}\exp(-\left\|\mathbf{z}_i^a - \mathbf{z}_k^b\right\|_1)}, \quad \beta_{ij} = \frac{\exp(-\left\|\mathbf{z}_i^a - \mathbf{z}_j^b\right\|_1)}{\sum_{k=1}^{n_1}\exp(-\left\|\mathbf{z}_k^a - \mathbf{z}_j^b\right\|_1)}.\end{gathered} \tag{5.3}$$

Intuitively, we can understand the soft-align comparison as computing an *expected alignment score*, where they are summed over all the possible alignments. We suppose that the smaller the distance between representations, the more likely it is that the pair of amino acids is aligned. Then, we can consider α_{ij} as the probability that $\mathbf{z}_i^a$ is aligned to $\mathbf{z}_j^b$, considering all the amino acids from Z^b (and vice versa for β_{ij}). As a result, $\hat{c}$ is the expected alignment score over all possible alignments with probabilities ω_{ij}. Note that the negative signs are applied for converting distances into scores. Therefore, a higher value of $\hat{c}$ indicates that the pair of protein sequences is structurally more similar.

Given the similarity score, the SFP output layer computes the probability that the pair belongs to the same protein family. The SFP task trains PLUS-RNN to minimize the cross-entropy loss between the true label and the predicted probability. As the model is designed to produce higher similarity scores for proteins from the same families, learned representations must convey global structural information.

5.1.2 Fine-tuning Procedure

The fine-tuning procedure of PLUS-RNN follows the conventional usage of BiRNN-based prediction models. For each downstream task, we add one hidden layer and one output layer on top of the pre-trained model. Then, all the parameters are fine-tuned using task-specific datasets. The complete fine-tuning loss is defined as:

$$\mathcal{L}_{\text{FT}} = \lambda_{\text{FT}}\mathcal{L}_{\text{MLM}} + (1 - \lambda_{\text{FT}})\mathcal{L}_{\text{TASK}} \tag{5.4}$$

where $\mathcal{L}_{\text{TASK}}$ is the task-specific loss. $\mathcal{L}_{\text{MLM}}$ is the regularization loss. We use λ_{FT} to control their relative importance.

The model's architectural modifications for the three types of downstream tasks are as follows. For tasks involving a protein pair, we use the same computations used in the SFP pre-training task. Specifically, we replace only the SFP

output layer with a new output layer. For single protein-level tasks, we adopt an additional attention layer to aggregate variable-length representations into a single vector [30]. Then, the aggregated vector is fed into the hidden and output layers. For amino-acid-level tasks, representations of each amino acid are fed into the hidden and output layers.

5.1.3 Model Architecture

PLUS can be used to pre-train various model architectures including BiRNN and the TFM. The resulting models are referred to as PLUS-RNN and PLUS-TFM, respectively. In this work, we mainly used PLUS-RNN, because of its two advantages over PLUS-TFM. First, it is more effective for learning the sequential nature of proteins. The self-attention layer of the TFM performs dot products between all pairwise tokens regardless of their positions within the sequence. In other words, it provides an equal opportunity for local and long-range contexts to determine the representations. Although this facilitates the learning of long-range dependencies, the downside is that it completely ignores the *locality bias* within a sequence. This is particularly problematic for protein biology, where local amino acid motifs often have significant structural and functional implications [106]. In contrast, RNN sequentially processes a sequence, and local contexts are naturally more emphasized.

Second, PLUS-RNN provides lower computational complexity. Although the model hyperparameters have an effect, the TFM-based models generally demand a larger number of parameters than RNNs [28]. Furthermore, the computations between all pairwise tokens in the self-attention layer impose a considerable computational burden, which scales quadratically with the input sequence length. Considering that pre-training typical TFM-based models handling 512 tokens already requires tremendous resources [99], it is computationally difficult to use TFMs to manage longer protein sequences, even up to a few thousand amino acids.

In this section, we explain the architecture of PLUS-RNN. First, an input embedding layer, EM, embeds each amino acid, $\mathbf{x}_i$, into a d_e-dimensional dense vector, $\mathbf{e}_i$:

$$E = [\mathbf{e}_1, \cdots, \mathbf{e}_n], \quad \mathbf{e}_i = \mathrm{EM}(\mathbf{x}_i). \tag{5.5}$$

Then, a BiRNN of L-layers computes representations as a function of the entire sequence. We use long short-term memory as the basic unit of the BiRNN [24]. In each layer, the BiRNN computes d_h-dimensional forward and backward hidden states ($\overrightarrow{\mathbf{h}_i^l}$ and $\overleftarrow{\mathbf{h}_i^l}$) and combines them into a hidden state, $\mathbf{h}_i^l$, using a non-linear transformation:

$$\begin{aligned} \overrightarrow{\mathbf{h}_i^l} &= \sigma(\overrightarrow{\mathbf{W}_x^l}\mathbf{h}_i^{l-1} + \overrightarrow{\mathbf{W}_h^l}\mathbf{h}_{i-1}^l + \overrightarrow{\mathbf{b}^l}), \\ \overleftarrow{\mathbf{h}_i^l} &= \sigma(\overleftarrow{\mathbf{W}_x^l}\mathbf{h}_i^{l-1} + \overleftarrow{\mathbf{W}_h^l}\mathbf{h}_{i+1}^l + \overleftarrow{\mathbf{b}^l}), \\ \mathbf{h}_i^l &= \sigma(\mathbf{W}_h^l[\overrightarrow{\mathbf{h}_i^l}; \overleftarrow{\mathbf{h}_i^l}] + \mathbf{b}^l) \quad for \quad l = 1, \cdots, L, \end{aligned} \tag{5.6}$$

where $\mathbf{h}_i^0 = \mathbf{e}_i$; $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias vectors, respectively. We use the final hidden states, $\mathbf{h}_i^L$, as representations, $\mathbf{r}_i$, of each amino acid:

$$R = [\mathbf{r}_1, \cdots, \mathbf{r}_n], \quad \mathbf{r}_i = \mathbf{h}_i^L. \tag{5.7}$$

We adopt an additional projection layer to obtain smaller d_z-dimensional representations $\mathbf{z}_i$ of each amino acid with a linear transformation:

$$Z = [\mathbf{z}_1, \cdots, \mathbf{z}_n], \quad \mathbf{z}_i = \mathrm{Proj}(\mathbf{r}_i). \tag{5.8}$$

During pre-training, to reduce computational complexity, we use R and Z for the MLM and SFP tasks, respectively. During fine-tuning, we can use either R or Z, considering the performance on development sets or based on the computational constraints.

We use two models with the fixed d_e of 21 and d_z of 100:

- PLUS-RNN$_{\mathrm{BASE}}$: $L = 3$, d_h= 512, 15M parameters

- PLUS-RNN$_{\text{LARGE}}$: $L = 3$, d_h= 1024, 59M parameters

The hyperparameters (*i.e.*, L and d_h) of PLUS-RNN$_{\text{BASE}}$ are chosen to match the BiRNN model architecture used in P-ELMo [17]. However, as P-ELMo uses additional RNNs, PLUS-RNN$_{\text{BASE}}$ has less than half the number of parameters that P-ELMo has (32M).

5.2 Experiment Results

5.2.1 Experiment Setup

Pre-training Dataset

We used Pfam (release 27.0) as the pre-training dataset [105]. After pre-processing, it contained 14,670,860 sequences from 3,150 families. The Pfam protein family was computationally constructed by comparing sequence similarity using alignments. Owing to the loose connection between sequence and structure similarities, the family labels only provide weak structural information [107]. Note that we did not use any human-annotated labels. Therefore, pre-training does not result in biased evaluations in fine-tuning tasks.

Fine-tuning Tasks

We evaluated PLUS-RNN on seven protein biology tasks. The datasets were curated and pre-processed by the cited studies. In the main manuscript, we provide concise task definitions and evaluation metrics.

Homology is a protein-level classification task [108]. The goal is to classify the structural similarity level of a protein pair into *family*, *superfamily*, *fold*, *class*, or *none*. We report the accuracy of the predicted similarity level and the Spearman correlation, ρ, between the predicted similarity scores and the true similarity levels. Furthermore, we provide the average precision (AP) from prediction scores at each similarity level.

Solubility is a protein-level classification task [109]. The goal is to predict whether a protein is *soluble* or *insoluble*. We report the accuracy of this task.

Localization is a protein-level classification task [52]. The goal is to classify a protein into one of 10 subcellular locations. We report the accuracy of this task.

Stability is a protein-level regression task [110]. The goal is to predict a real-valued proxy for intrinsic stability. This task is from TAPE [19], and we

report the Spearman correlation, ρ.

Fluorescence is a protein-level regression task [111]. The goal is to predict the real-valued fluorescence intensities. This task is from TAPE, and we report the Spearman correlation, ρ.

Secondary structure (SecStr) is an amino-acid-level classification task [112]. The goal is to classify each amino acid into eight or three classes, that describe its local structure. This task is from TAPE. We report both the three-way and eight-way classification accuracies (Q8/Q3) of this task.

Transmembrane is an amino-acid-level classification task [113]. The goal is to detect amino acid segments that cross the cell membrane. We report the accuracy of this task.

Baselines

We provided several baselines for comparative evaluations. Note that since up-scaling of models and datasets often provide performance improvements, we only considered those with a similar scale of model sizes and pre-training datasets to focus on evaluating the pre-training methods.

First, in all the tasks, we used two baselines: P-ELMo and PLUS-TFM. The former has a model architecture similar to PLUS-RNN$_{\text{BASE}}$; thus, it can show the effectiveness of the pre-training method. The latter is pre-trained with PLUS, so it can show the effectiveness of the BiRNN compared to the TFM architecture.

Second, for the tasks from TAPE, we provide their reported baselines: P-ELMo, UniRep, TAPE-TFM, TAPE-RNN, and TAPE-ResNet. Note that these comparisons are in their favor, as they used a larger pre-training dataset (32M proteins from Pfam release 32.0). The TAPE baselines can demonstrate that PLUS-RNN outperforms models of similar size solely pre-trained with the LM.

Finally, we benchmarked PLUS-RNN against task-specific SOTA models

Table 5.1: Results on pre-training tasks.

Method	(M)LM (acc)	SFP (acc)
PLUS-TFM	0.37	**0.98**
PLUS-RNN_{BASE}	0.33	0.96
PLUS-$\text{RNN}_{\text{LARGE}}$	0.37	0.97
P-ELMo∗	0.29	-
P-ELMo†	0.28	-
UniRep†	0.32	-
TAPE-TFM†	**0.45**	-
TAPE-RNN†	0.40	-
TAPE-ResNet†	0.41	-

∗Our experiments (Pfam 27.0). †Excerpted from TAPE (Pfam 32.0).

trained from scratch. If no deep learning-based baseline exists for a given task, we provided RNN_{BASE} and $\text{RNN}_{\text{LARGE}}$ models without pre-training. The comparison with those exploit additional features can help us identify the tasks for which the proposed pre-training method is most effective and help us understand its current limitations.

5.2.2 Pre-training Results

Table 5.1 lists the test accuracies for the MLM and SFP pre-training tasks. Only the models pre-trained with PLUS were evaluated for the SFP task. Note that our experiments and TAPE used different test datasets; care should be taken in comparing them. Nonetheless, we can still indirectly compare them, considering the following. First, both the test datasets comprised randomly sampled proteins from different versions of the Pfam dataset (27.0 for PLUS and 32.0 for TAPE). Second, P-ELMo was evaluated in both datasets and showed similar LM accuracies. This indicates that the difference between the two datasets is negligible.

We can see that some models have lower LM accuracies than others. However, the lower LM capability does not precisely correspond to the performance in fine-tuning tasks. This discrepancy has been previously observed in TAPE,

and it can also be observed in the following sections. In terms of SFP, all the models pre-trained with PLUS exhibited high accuracies. As the Pfam families were constructed based only on sequence similarities, a pair of analogous sequences would probably be from the same family. Despite its simplicity, we empirically demonstrated that the SFP complements the MLM by encouraging the models to compare protein representations during pre-training.

5.2.3 Fine-tuning Results

Summarized Results

Table 5.2 presents the summarized results for the benchmark tasks. Specifically, we show the best results from two categories: LM pre-trained models and task-specific SOTA models.

The PLUS-RNN$_{\text{LARGE}}$ model outperformed models of similar size solely pre-trained with the conventional LM in six out of seven tasks. Considering that some pre-trained models exhibited higher LM capabilities, it can be speculated that the protein-specific SFP pre-training task contributed to the improvement. In the ablation studies, we further explained the relative importance of each aspect of PLUS-RNNȦlthough PLUS-TFM had almost twice as many parameters as PLUS-RNN$_{\text{LARGE}}$ (110M vs. 59M), it exhibited inferior performance in most tasks. We infer that this is because it disregarded the *locality bias*.

We compared PLUS-RNN$_{\text{LARGE}}$ with task-specific SOTA models. Although the former performed better in some tasks, it still lagged behind on the others. The results indicated that tailored models with additional features provide powerful advantages that could not be learned through pre-training. A classic example is the use of position-specific scoring matrices generated from multiple sequence alignments. We conjectured that simultaneous observation of multiple proteins could facilitate evolutionary information. In contrast, current pre-training methods use millions of proteins; however, they still consider

Table 5.2: Fine-tuning results on protein biology benchmark task.

	Protein-level Classification		
Method	Homology (acc)	Solubility (acc)	Localization (acc)
PLUS-TFM	0.96	**0.72**	0.69
PLUS-RNN$_{\text{BASE}}$	0.96	0.70	0.69
PLUS-RNN$_{\text{LARGE}}$	**0.97**	0.71	**0.70**
LM Pre-trained	0.95	0.64	0.54
Task-specific SOTA	0.93	0.77	0.78

	Protein-level Regression		Amino-acid-level Classification	
Method	Stability (ρ)	Fluorescence (ρ)	SecStr (acc)	Transmembrane (acc)
PLUS-TFM	0.76	0.63	0.59	0.82
PLUS-RNN$_{\text{BASE}}$	**0.77**	0.67	0.61	**0.89**
PLUS-RNN$_{\text{LARGE}}$	**0.77**	**0.68**	**0.62**	**0.89**
LM Pre-trained	0.73	**0.68**	0.61	0.78
Task-specific SOTA	0.73	0.67	0.72	0.80

For each task, the best pre-trained model is in **bold**. It is **bold and underlined** if it is the best when including the task-specific SOTA.

Table 5.2: Detailed Homology prediction results.

	Overall		Per-level AP			
Method	acc	ρ	Class	Fold	Superfamily	Family
PLUS-TFM	0.96	**0.70**	0.94	0.91	0.95	**0.67**
PLUS-RNN$_{\text{BASE}}$	0.96	0.69	0.94	0.90	0.94	0.66
PLUS-RNN$_{\text{LARGE}}$	**0.97**	**0.70**	**0.95**	**0.92**	**0.96**	0.66
P-ELMo	0.95	0.69	0.90	0.88	0.94	0.65
P-ELMo†	0.95	0.69	0.91	0.90	0.95	0.65
NW-align†	0.78	0.22	0.31	0.41	0.58	0.53
phmmer†	0.78	0.07	0.31	0.26	0.35	0.54
HHalign†	0.79	0.23	0.40	0.62	0.86	0.52
TMalign†	0.81	0.37	0.55	0.85	0.83	0.57
RNN$_{\text{BASE}}$	0.93	0.66	0.86	0.80	0.89	0.62
RNN$_{\text{LARGE}}$	0.83	0.52	0.66	0.46	0.52	0.39

†Excerpted from P-ELMo.

each one individually. The relatively small improvement from PLUS could also be explained by the fact that the SFP task only utilizes pairwise information. We expect that investigating multiple proteins during pre-training might be the key to a superior performance over the task-specific SOTA models

Detailed Homology and SecStr results

We present detailed evaluation results for the Homology and SecStr tasks. We chose these two tasks because they are representative protein biology tasks relevant to global and local structures, respectively. Improved results on the former can lead to the discovery of new enzymes and antibiotic-resistant genes [114]. The latter is important for understanding the function of proteins in cases where evolutionary structural information is not available [112].

The detailed Homology prediction results are listed in Table 5.2. The results show that PLUS-RNN$_{\text{LARGE}}$ outperformed both P-ELMo and task-specific models. In contrast to RNN$_{\text{LARGE}}$, which exhibited overfitting owing to the limited labeled training data, PLUS pre-training enabled us to take advan-

Table 5.3: Detailed SecStr prediction results.

	CB513		CASP12		TS115	
Method	Q8	Q3	Q8	Q3	Q8	Q3
PLUS-TFM	0.59	0.73	0.57	0.71	0.65	0.77
PLUS-RNN$_{\text{BASE}}$	0.61	0.75	0.60	0.72	0.66	0.78
PLUS-RNN$_{\text{LARGE}}$	**0.62**	**0.77**	**0.60**	**0.73**	**0.68**	**0.79**
P-ELMo∗	0.61	**0.77**	0.54	0.68	0.63	0.76
P-ELMo†	0.58	0.73	0.57	0.70	0.65	0.76
UniRep†	0.57	0.73	0.59	0.72	0.63	0.77
TAPE-TFM†	0.59	0.73	0.59	0.71	0.64	0.77
TAPE-RNN†	0.59	0.75	0.57	0.70	0.66	0.78
TAPE-ResNet†	0.58	0.75	0.58	0.72	0.64	0.78
NetSurfP-2.0‡	0.72	0.85	0.70	0.82	0.75	0.86

†Excerpted from TAPE. ‡Excerpted from [112].

tage of the large model architecture. The correlation differences among PLUS-RNN$_{\text{LARGE}}$ (0.697), PLUS-RNN$_{\text{BASE}}$ (0.693), and P-ELMo (0.685) were small; however, they were statistically significant with p-values lower than 10^{-15} [115]. The per-level AP results helped us further examine the level of structural information captured by the pre-training. The largest performance improvement of PLUS-RNN comes at the higher *class* level rather than the lower *family* level. This indicates that even though Pfam family labels tend to be structurally correlated with the Homology task *family* levels [97], they are not the decisive factors for performance improvement. Instead, PLUS pre-training incorporates weak structural information and facilitates inferring higher-level global structure similarities.

The detailed SecStr prediction results are listed in Table 5.3. CB513, CASP12, and TS115 denote the SecStr test datasets. The results show that PLUS-RNN$_{\text{LARGE}}$ outperformed all the other models of similar size pre-trained solely with LM. This demonstrates that the SFP task complements the LM task during pre-training and helps in learning improved structurally contextualized representations. However, PLUS-RNN$_{\text{LARGE}}$ still lagged behind task-

Table 5.4: Pre-trained protein language models.

	Model	Parameter (M)	Dimensions	Proteins (M)
UniRep	RNN	18	1,900	24
PLUS-RNN	RNN	59	2,048	15
SeqVec	RNN	94	1,024	33
ProtXLNet	TFM	409	1,024	216
ProtBERT	TFM	421	1,024	2,122
ESM	TFM	669	1,280	27

specific SOTA models that employ alignment-based features. We infer that this limitation might be attributable to the following two factors. First, as previously stated, PLUS only utilizes pairwise information, rather than simultaneously examining multiple proteins during pre-training. Second, the SFP task requires an understanding of the global structures, and thus the local structures are relatively negligible. Therefore, we believe that devising an additional pre-training task relevant to local structural information would improve the performance on the SecStr task.

5.2.4 Comparison with Larger Protein Language Models

We compare PLUS with larger protein LMs using the HSP identification task. The key differences among the protein LMs originate from two factors: (1) LM architecture and (2) the number of proteins used for pre-training (Table 5.4). In terms of the LM architecture, UniRep, PLUS-RNN, and SeqVec use RNNs; ProtXLNet, ProtBERT, and ESM use TFMs. RNN-based models require less resources for both pre-training and producing representations. Although TFM-based models require significantly more resources, they are better at capturing long-term dependencies within proteins and can provide more informative representations [28]. The number of unlabeled proteins used in each study varied considerably. The LMs with more parameters were usually pre-trained with a larger number of proteins. Exceptionally, while ESM used the largest protein

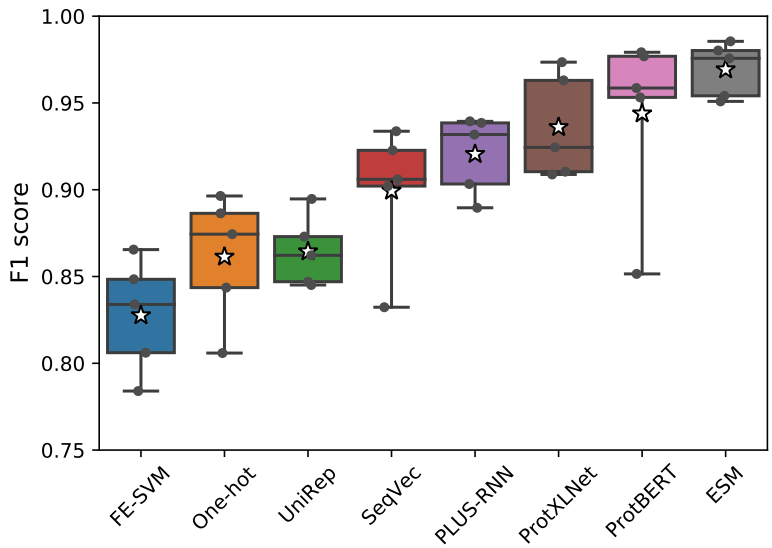

Figure 5.2: HSP identification results using different protein LMs.

LM, it was pre-trained with a relatively small number of proteins. This can be attributed to its high-diversity dataset, which contains only representative proteins from clusters based on sequence identity [103].

HSPs are stress-induced proteins that are highly conserved across organisms ranging from bacteria to humans. They play a pivotal role as molecular chaperones against unfavorable conditions, such as elevated temperature and inflammation. According to core functions and molecular weights [116], HSPs can be categorized into six major families. Previous studies on the computational identification of HSP families have two major limitations. First, they relied heavily on amino acid composition features, which inevitably limited their performance. Second, their prediction performance was overestimated because of the independent two-stage evaluation and data redundancy [117, 118].

We trained a CNN model on top of pre-trained protein representations. We explored various pre-trained protein LMs to compare their effectiveness: UniRep, SeqVec, PLUS-RNN, ProtXLNet, ProtBERT, and ESM. Given a protein sequence, we used a pre-trained protein LM to convert it into a sequence of representations. Then, a projection layer with shared weights across different

positions embeds them into 20-dimensional vectors. The CNN model consists of a convolution layer, a global max-pooling layer, and a fully-connected layer. The global max-pooling layer computes the maximum value of the output of each filter to obtain a fixed-length representation vector. Finally, the fully-connected layer computes the classification outputs.

For cross-validation experiments, we utilized the same dataset used in previous studies [117, 118]. Non-HSP sequences were randomly selected without homologous proteins from SwissProt [119]. HSP sequences were derived from HSPIR [116]. Thereafter, the proteins with $\geq 40\%$ pairwise sequence similarity within the same family were removed using CD-HIT [120]. Finally, the non-HSP and HSP sequences containing non-standard amino acids were filtered out to obtain a cross-validation dataset.

We compared the performance of using different protein LMs through five-fold cross-validation (Figure 5.2). As a baseline, we include the performance of SOTA feature extraction-based SVM (FE-SVM) model and a CNN model with one-hot encoding. Each box denotes the quartiles of F1 scores, and the star denotes their average. The boxplot shows that all LMs improve the average classification performance compared to the one-hot encoding and the feature extraction-based SOTA SVM model. Taking advantage of a large number of unlabeled proteins, the pre-trained protein representations provide a wealth of information that cannot be learned from one-hot encoding.

While all the pre-trained protein LMs help in the identification of HSP families, their level of performance improvement varies significantly. The small gap between OneHot and UniRep indicates that a sufficient number of parameters are required to obtain a moderate increase (Table 5.4). LMs with more parameters generally provide more performance improvement. For example, the larger TFM-based LMs outperformed the RNN-based LMs, and the largest ESM showed the best performance. One exception is that although PLUS-RNN has fewer parameters than SeqVec, it exhibits better performance. We

conjecture that this can be attributed to its additional protein-specific pre-training objective, which can better capture structural information of protein sequences than those solely pre-trained with an LM.

5.2.5 Ablation Studies

Here, we show results from ablation studies on the Homology and SecStr tasks to better understand the strengths and aspects of the PLUS framework. We used PLUS-RNN$_{\text{BASE}}$ as the baseline model unless explicitly stated otherwise. Note that we used the development sets for the ablation studies.

Pre-training and Fine-tuning of PLUS-RNN

We explored the effect of using different λ_{PT} values for controlling the relative importance of the MLM and SFP pre-training tasks (Table 5.5). The results indicated that the pre-trained models with different λ_{PT} values (PLUS-RNN$_{\text{BASE}}$, PT-A, PT-B, PT-C) always outperformed the RNN$_{\text{BASE}}$ model trained from the scratch. Both pre-training tasks consistently improve the prediction performance at all structural levels. Of the two pre-training tasks, removing MLM negatively affects the prediction performance more than removing the SFP. This coincides with the expected result, according to which, the MLM task would play the primary role, and the SFP task would complement MLM by encouraging the models to compare pairwise representations.

During the fine-tuning, we simultaneously trained a model for the MLM task as well as the downstream task. Moreover, we explored the effect of using different λ_{FT} values for controlling their relative importance (Table 5.5). The results showed that the models simultaneously fine-tuned with the MLM task loss (PLUS-RNN$_{\text{BASE}}$ and FT-B) consistently outperformed the (FT-A) model fine-tuned only with the task-specific loss. Based on this, we infer that the MLM task serves as a form of regularization and improves the generalization performance of the models.

Table 5.5: Ablation studies on Homology and SecStr tasks.

			Homology						SecStr	
Method	λ_{PT}	λ_{FT}	acc	ρ	Class	Fold	Superfamily	Family	acc8	acc3
PLUS-RNN$_{\text{BASE}}$	0.7	0.3	**0.96**	**0.70**	**0.95**	**0.91**	**0.96**	**0.72**	**0.66**	**0.78**
RNN$_{\text{BASE}}$	-	0.3	0.93	0.67	0.88	0.81	0.92	0.68	0.61	0.73
(PT-A)	0.0	0.3	0.94	0.68	0.91	0.85	0.93	0.70	0.62	0.74
(PT-B)	0.5	0.3	**0.96**	0.69	**0.95**	**0.91**	0.95	0.70	0.66	0.77
(PT-C)	1.0	0.3	**0.96**	0.69	0.93	0.89	0.95	0.70	0.65	0.77
(FT-A)	0.7	0.0	0.94	0.68	0.91	0.85	0.93	0.70	0.65	0.77
(FT-B)	0.7	0.5	**0.96**	0.69	**0.95**	**0.91**	0.95	0.70	**0.66**	**0.78**

We use the development sets for the ablation studies.

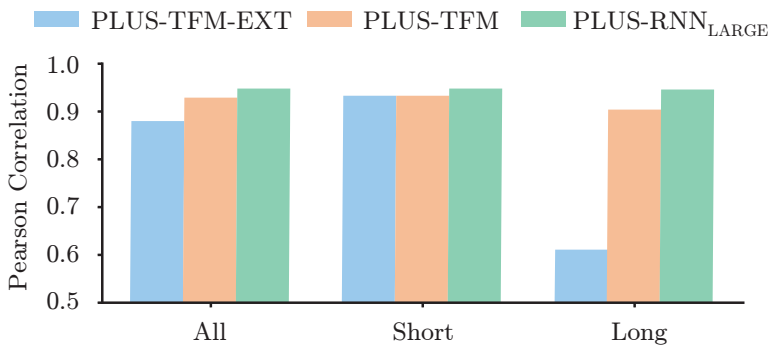

Figure 5.3: Homology prediction results for different lengths.

Comparison of PLUS-RNN and PLUS-TFM

We compared the Homology prediction performances of PLUS-TFM and PLUS-RNN$_{\text{LARGE}}$ for protein pairs of different lengths (Figure 5.3). Because PLUS-TFM was pre-trained using protein pairs shorter than 512 amino acids, we denote *Long* for protein pairs longer than 512 amino acids and *Short* otherwise. Next, we evaluated PLUS-TFM for the *Long* protein pairs in the following two ways. First, we simply used the protein pairs as they were. Second, we truncated them to 512 amino acids. The former is denoted as PLUS-TFM-EXT (as in extended), and the latter is denoted as PLUS-TFM.

PLUS-RNN$_{\text{LARGE}}$ consistently provided competitive performance regardless of the protein length. In contrast, PLUS-TFM-EXT deteriorated for the *Long* protein pairs, whereas PLUS-TFM exhibited a relatively less performance degradation. The results presented the limitations of TFM models using the limited context size of 512 amino acids. Although the number of *Long* protein pairs in the Homology development dataset was relatively small (13.4%), complex proteins that are found in nature make the ability to analyze long protein sequences indispensable. Moreover, because this is due to the computational burden of TFM scaling quadratically with the input length, we predict that the recently proposed adaptive attention span approach [121] may be able to help improve PLUS-TFM.

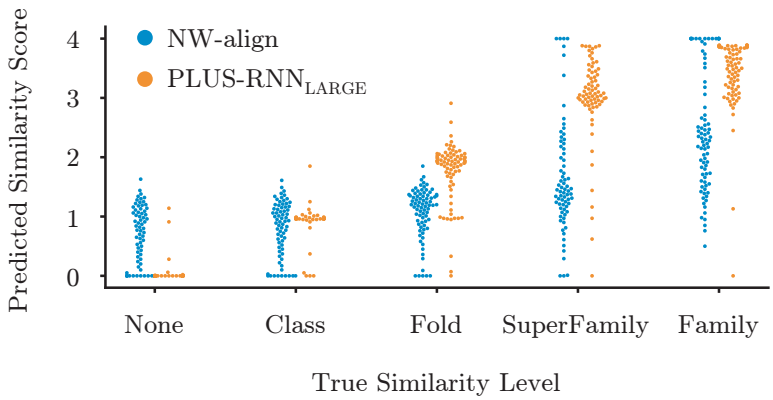

Figure 5.4: Plot of predicted similarity scores and true similarity levels. Results from NW-align (left) and PLUS-RNN$_{\text{LARGE}}$ (right) are presented in each similarity level. Figure best viewed in color.

5.2.6 Qualitative Interpreatation Analyses

To better understand the strengths of PLUS pre-training, we provide its qualitative analyses. We examined the Homology task to interpret how the learned protein representations help infer the global structural similarities of proteins

To compare two proteins, PLUS-RNN used soft-align to compute a similarity score, $\hat{c}$. Even though there was one more computation by the output layer for the Homology prediction output, we could use the similarity scores to interpret PLUS-RNN. Note that using the penultimate layer for model interpretation is widely adopted in the machine learning community [122].

Figure 5.4 shows a scatter plot of the predicted similarity scores and true similarity levels. For comparison, we also show the NW-align results based on the BLOSUM62 scoring matrix [95]. The plot shows that NW-align often produces low similarity scores for protein pairs from the same *family*. This is because of high sequence-level variations, which result in dissimilar sequences having similar structures. In contrast, PLUS-RNN$_{\text{LARGE}}$ produces high similarity scores for most protein pairs from the same *family*.

Furthermore, we examined three types of protein pairs: (1) a similar sequence-similar structure pair, (2) a dissimilar sequence-similar structure pair, and (3)

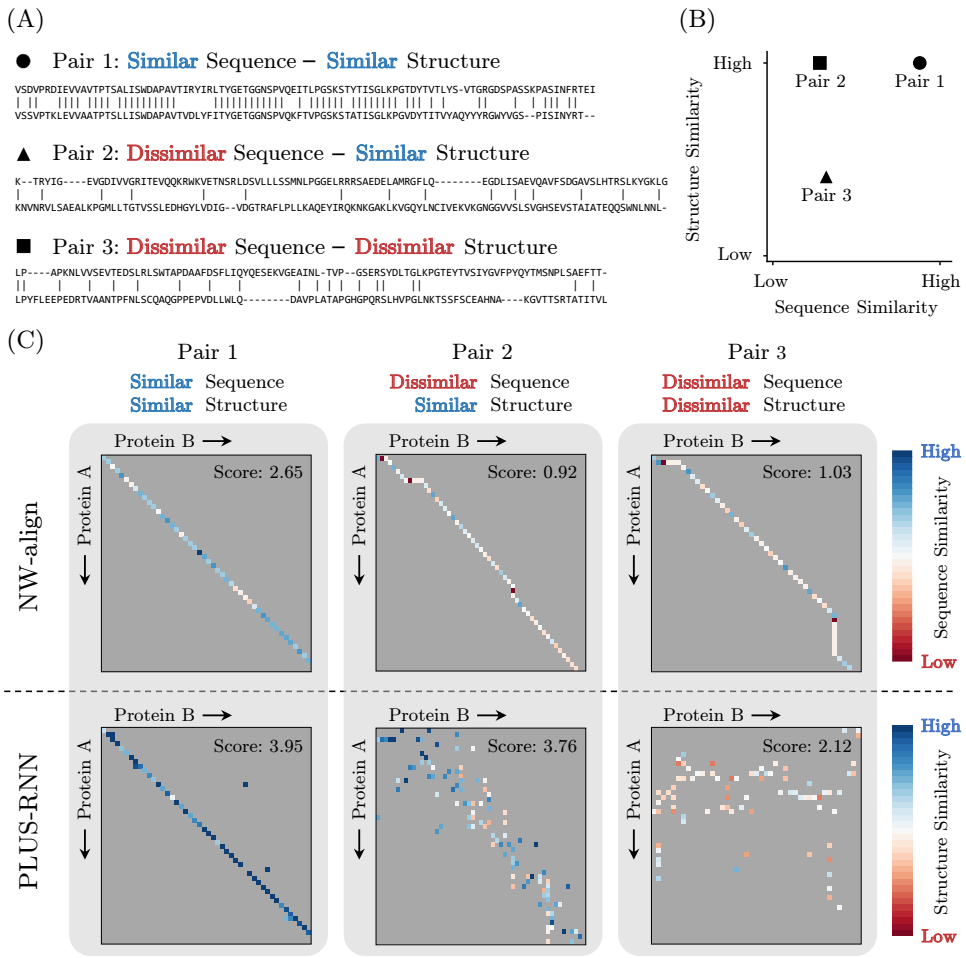

Figure 5.5: Interpretation of predictions for Homology task. (A) We investigate three types of protein pairs: a similar sequence - similar structure pair, a dissimilar sequence - similar structure pair, and a dissimilar sequence - dissimilar structure pair. (B) The sequence and structure similarities of each pair are defined by NW-align scores and Homology dataset labels, respectively. (C) Heatmaps of NW-align of raw amino acids and the soft-alignment of PLUS-RNN representations for the three pairs. Figure best viewed in color.

a dissimilar sequence-dissimilar structure pair (Figure 5.5(A) and (B)). Note that similar sequence-dissimilar structure pairs did not exist in the Homology datasets. The sequence and structure similarities were defined by NW-align scores and Homology dataset labels, respectively. The pairs with similar structures were chosen from the same *family*, and those with dissimilar structures were chosen from the same *fold*. Figure 5.5(C) shows the heatmaps of the NW-align of raw amino acids and soft-alignment of PLUS-RNN representations (ω_{ij}) for the three pairs. Owing to space limitations, we only show the top left quadrant of the heatmaps. Each cell in the heatmap indicates the corresponding amino acid pairs from proteins A and B. Blue denotes high sequence similarity in NW-align and high structure similarity in PLUS-RNN.

First, we compared the pairs having similar structures (the first and second columns in Figure 5.5(C)). The heatmaps show that NW-align successfully aligned the similar-sequence pair, resulting in a score of 2.65. However, it failed for the dissimilar-sequence pair, with a score of 0.92. This supports the observation that comparing raw sequence similarities cannot identify the correct structural similarities. In contrast, the soft-alignment of PLUS-RNN representations was successful for both similar and dissimilar sequences, with scores of 3.95 and 3.76, respectively. Next, we compared the second and third pairs. Although only the second pair had similar structures, NW-align failed for both and even yielded a higher score of 1.03 for the third pair. In contrast, regardless of the sequence similarities, the soft-alignment of PLUS-RNN representations correctly decreased only for the third pair, with dissimilar structures having a score of 2.12. Therefore, the interpretation results confirmed that the learned representations from PLUS-RNN are structurally contextualized and perform better in inferring global structure similarities.

5.3 Summary

We presented PLUS, a novel pre-training method for bidirectional protein sequence representations. Consisting of the MLM and protein-specific SFP pre-training tasks, PLUS outperformed the conventional LM pre-training methods by capturing structural information contained within the proteins. PLUS can be used to pre-train various model architectures. In this work, we used PLUS-RNN because of its superior sequential modeling capability and lower computational complexity. PLUS-RNN outperformed models of similar size solely pre-trained with the conventional LM in six out of seven protein biology tasks. To better understand its strengths, we also provided the results from our qualitative interpretation analyses.

We expect that the gap between the numbers of unlabeled and labeled proteins will continue to grow exponentially, and pre-training methods will play a larger role. We plan to extend this work in several directions. First, considering that PLUS-RNN is powerful for inferring global structural information, we are interested in a more refined prediction of protein structures [123]. Second, although pre-training helps, our method still lags behind task-specific models in some tasks. We think that this limitation comes from weaknesses in learning evolutionary information. We believe that there is still considerable room for improvement. Investigation of multiple proteins during pre-training, as in the alignment, could be the key [124].

Chapter 6

Discussion

6.1 Challenges and Opportunities

In this dissertation, we proposed a set of representation learning methods to analyze biological sequence data. While they have made significant contributions to different problems in bioinformatics, there is obviously huge room for improvement. In the following, we will discuss several other challenges and opportunities for future work.

One of the fundamental challenges in biological sequence analyses is capturing both short-term and long-term dependencies. In biological sequences, local discriminative patterns such as regulatory motifs definitely hold great significance. Nevertheless, biological sequences are often extremely long, and elements in distant positions can also have close dependencies. For example, proteins naturally fold into three-dimensional structures determined by their amino acid sequences. As these structures have a direct impact on protein functions, amino acids in distant positions often co-mutate to maintain the indispensable structures [57]. Therefore, in order to comprehensively analyze and understand biological sequences, it is crucial to capture both short-term and long-term dependencies. There are two research directions for capturing both short-term and long-term dependencies: the development of (1) DNN

architectures and (2) training algorithms. With abundant training data, it is possible and even beneficial to learn any dependencies solely from data. However, obtaining millions of sequence data is often difficult due to complex and expensive data acquisition processes in the bioinformatics domain. For smaller datasets, introducing proper inductive biases into the DNN architectures or training with more explicit information can facilitate learning both types of dependencies more easily [125]. We think some of the promising approaches would come from hybrid transformer models accompanied by convolutions [126] and training with co-evolutionary information from multiple sequence alignments [127].

The successes of representation learning algorithms in image recognition and natural language processing did not solely come from the advanced DNN architectures. Training procedure refinements have also significantly contributed to their improved performance [128]. On the other hand, training procedure refinements for biological sequence analyses have not been studied thoroughly, which leaves significant room for expansion and innovation. We believe some of the research opportunities are in the following directions. First, data augmentations, which generate semantically similar but seemingly different data, have played crucial roles in computer vision tasks. They are a cornerstone not only to regularize large models effectively but also to learn representations from unlabeled data with self-supervised contrastive learning algorithms [129]. Since data augmentations heavily rely on domain-specific knowledge, proper augmentation strategies for biological sequences are still quite difficult and under-developed. Another challenge and opportunity are in learning under positive-negative data imbalance, where the number of negative samples is significantly larger than the number of positive samples. For example, in disease-related research, treatment groups are often smaller than the normal (control) groups due to their rareness and privacy restriction. The positive-negative data imbalance often leads to a poor model performance by under-

emphasizing the positive samples during training. Currently, the most widely used approaches to deal with the imbalance are data sampling and constant class-weighted objective loss. The former balances the class distribution based on methods such as the synthetic minority oversampling technique [130]. The latter explicitly underrate training losses for negative samples with pre-defined weights. We believe biological sequence analyses will be benefited more from advanced methods such as dynamic down-weights for easy negative samples [131] and dynamically generating difficult-to-classify positive samples based on generative adversarial networks [132].

While representation learning has enabled unprecedented breakthroughs in various domains, it has been mainly focused on improving its prediction performance. However, merely providing good outcomes is not enough anymore [7]. Researchers are also interested in how they would impact society and have started to focus on different aspects. One of the major concerns is that representation learning algorithms are generally used as a black-box. We still know very little about how DNNs understand the world for producing predictions. Interpretability matters, particularly concerning bioinformatics and healthcare. To build trust in the algorithms, we need to make sure to provide logical reasoning behind its predictions like clinicians do for medical treatments. Approaches to interpreting DNNs can be categorized into two groups *i.e.*, local explanations and global explanations. Local explanation methods aim to interpret model predictions for each sample. Global explanation methods aim to interpret how a model characterizes each class of samples. For example, DeepBind presented two interpretation approaches for transcription factor binding site prediction [46]. First, the authors proposed a mutation map for local explanations. A mutation man illustrates the importance of each nucleotide by measuring the maximum change of prediction scores among all possible mutations. Second, the authors proposed to use motifs for global explanations of a trained model. For each filter in the convolution layer, they

aligned all the subsequences which passed the activation threshold. Then, they generated a position-specific scoring matrix and transformed it into a sequence logo representing a motif. We believe interest in demystifying DNNs for biological sequence analyses will continue to grow. They will help us not only to understand models based on our current knowledge but also to acquire new biological knowledge from them.

Chapter 7

Conclusion

While representation learning has shown great promise in diverse fields, it is not a silver bullet. Full of challenges remain to derive invaluable information through bioinformatics research. This dissertation proposed a set of representation methodologies to address the three issues for biological sequence data analysis. The contributions of this dissertation are summarized as follows:

- In Chapter 3, we addressed throughput and information trade-offs within wet-lab experiments. We proposed a two-stage strategy to train a deep-learning framework that can consider both genetic and epigenetic factors for CRISPR-Cpf1 activity prediction. First, we pre-trained a model with the integrated target dataset for the target sequence composition. Then. We fine-tuned the model with the endogenous target dataset to integrate chromatin accessibility. The proposed DeepCpf1 model showed significant performance boosts both in-house and independent datasets with an unprecedented level of high accuracy.

- In Chapter 4, we addressed modeling interaction between two sequences for functional microRNA target prediction. We proposed an encoding scheme to incorporate sequence alignment information. It represents not only the miRNA-CTS sequences but also how pairings, mismatches, or

gaps are formed within their extended seed regions. Combined with the relaxed CTS selection criteria and ResNet prediction model, the proposed TargetNet demonstrated significant classification performance improvement. Its top-ranked prediction scores showed a high association with the level of miRNA-mRNA expression down-regulation.

- In Chapter 5, we addressed the exponentially growing gap between the numbers of unlabeled and labeled protein sequences. We proposed PLUS, a novel pre-training method for bidirectional protein sequence representations. Consisting of the MLM and protein-specific SFP pre-training tasks, PLUS can capture structural information contained within unlabeled protein sequences. Our quantitative experiments and qualitative interpretation analyses demonstrated that PLUS-RNN outperformed models of similar size solely pre-trained with the conventional LM in six out of seven widely used protein biology tasks.

This dissertation lies at the intersection of representation learning and bioinformatics. We believe representation learning will be able to reveal the secrets behind genetic information, formulate effective therapeutics, and eventually revolutionize the healthcare system. To this end, we envision this dissertation to be a part of a disciplined and consistent effort to realize personalized medicine, one of the long-time goals in bioinformatics research.

Bibliography

[1] P. E. Bourne, V. Bonazzi, M. Dunn, E. D. Green, M. Guyer, G. Komatsoulis, J. Larkin, and B. Russell, "The nih big data to knowledge (bd2k) initiative," *Journal of the American Medical Informatics Association*, vol. 22, no. 6, pp. 1114–1114, 2015.

[2] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafé, A. Pérez, *et al.*, "Machine learning in bioinformatics," *Briefings in bioinformatics*, vol. 7, no. 1, pp. 86–112, 2006.

[3] M. W. Libbrecht and W. S. Noble, "Machine learning applications in genetics and genomics," *Nature Reviews Genetics*, vol. 16, no. 6, pp. 321–332, 2015.

[4] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular systems biology*, vol. 12, no. 7, p. 878, 2016.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in bioinformatics*, vol. 18, no. 5, pp. 851–869, 2017.

[8] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[9] J. G. Doench, E. Hartenian, D. B. Graham, Z. Tothova, M. Hegde, I. Smith, M. Sullender, B. L. Ebert, R. J. Xavier, and D. E. Root, "Rational design of highly active sgrnas for crispr-cas9–mediated gene inactivation," *Nature biotechnology*, vol. 32, no. 12, pp. 1262–1267, 2014.

[10] Y. Kim, S.-A. Cheong, J. G. Lee, S.-W. Lee, M. S. Lee, I.-J. Baek, and Y. H. Sung, "Generation of knockout mice by cpf1-mediated gene targeting," *Nature biotechnology*, vol. 34, no. 8, pp. 808–810, 2016.

[11] H. K. Kim, M. Song, J. Lee, A. V. Menon, S. Jung, Y.-M. Kang, J. W. Choi, E. Woo, H. C. Koh, J.-W. Nam, *et al.*, "In vivo high-throughput profiling of crispr–cpf1 activity," *Nature methods*, vol. 14, no. 2, pp. 153–159, 2017.

[12] J. G. Doench, N. Fusi, M. Sullender, M. Hegde, E. W. Vaimberg, K. F. Donovan, I. Smith, Z. Tothova, C. Wilen, R. Orchard, *et al.*, "Optimized sgrna design to maximize activity and minimize off-target effects of crispr-cas9," *Nature biotechnology*, vol. 34, no. 2, pp. 184–191, 2016.

[13] D. P. Bartel, "Micrornas: target recognition and regulatory functions," *cell*, vol. 136, no. 2, pp. 215–233, 2009.

[14] F. Kern, C. Backes, P. Hirsch, T. Fehlmann, M. Hart, E. Meese, and A. Keller, "What's the target: understanding two decades of in silico microrna-target prediction," *Briefings in Bioinformatics*, 2019.

[15] 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.

[16] L. Holm and C. Sander, "Mapping the protein universe," *Science*, vol. 273, no. 5275, pp. 595–602, 1996.

[17] T. Bepler and B. Berger, "Learning protein sequence embeddings using information from structure," in *International Conference on Learning Representations*, p. ., 2019.

[18] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, "Unified rational protein engineering with sequence-based deep representation learning," *Nature methods*, vol. 16, no. 12, pp. 1315–1322, 2019.

[19] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating protein transfer learning with tape," in *Advances in neural information processing systems*, pp. 9689–9701, 2019.

[20] F.-H. Hsu, *Behind Deep Blue: Building the computer that defeated the world chess champion.* Princeton University Press, 2002.

[21] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[23] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

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

[25] C. Olah, "Understanding lstm networks," Aug 2015.

[26] 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.

[27] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

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

[31] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

[32] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

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

[34] I. Loshchilov and F. Hutter, "Sgdrr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[37] 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.

[38] Y. Chen, Y. Li, R. Narayan, A. Subramanian, and X. Xie, "Gene expression inference with deep learning," *Bioinformatics*, vol. 32, no. 12, pp. 1832–1839, 2016.

[39] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, "A deep learning method for classification of eeg data based on motor imagery," in *International Conference on Intelligent Computing*, pp. 203–210, Springer, 2014.

[40] R. Heffernan, K. Paliwal, J. Lyons, A. Dehzangi, A. Sharma, J. Wang, A. Sattar, Y. Yang, and Y. Zhou, "Improving prediction of secondary structure, local backbone angles and solvent accessible surface area of proteins by iterative deep learning," *Scientific reports*, vol. 5, no. 1, pp. 1–11, 2015.

[41] T. Lee and S. Yoon, "Boosted categorical restricted boltzmann machine for computational prediction of splice junctions," in *International conference on machine learning*, pp. 2483–2492, PMLR, 2015.

[42] S. Lee, M. Choi, H.-s. Choi, M. S. Park, and S. Yoon, "Fingernet: Deep learning-based robust finger joint detection from radiographs," in *2015*

IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 1–4, IEEE, 2015.

[43] Y. Jeon, S. Kim, H.-S. Choi, Y. G. Chung, S. A. Choi, H. Kim, S. Yoon, H. Hwang, and K. J. Kim, "Pediatric sleep stage classification using multi-domain hybrid neural networks," *IEEE Access*, vol. 7, pp. 96495–96505, 2019.

[44] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with deep neural networks," *Medical image analysis*, vol. 35, pp. 18–31, 2017.

[45] Y. Park and M. Kellis, "Deep learning for regulatory genomics," *Nature biotechnology*, vol. 33, no. 8, pp. 825–826, 2015.

[46] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of dna-and rna-binding proteins by deep learning," *Nature biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[47] D. R. Kelley, J. Snoek, and J. L. Rinn, "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks," *Genome research*, vol. 26, no. 7, pp. 990–999, 2016.

[48] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature methods*, vol. 12, no. 10, pp. 931–934, 2015.

[49] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri, "Exploiting the past and the future in protein secondary structure prediction," *Bioinformatics*, vol. 15, no. 11, pp. 937–946, 1999.

[50] S. Park, S. Min, H.-S. Choi, and S. Yoon, "Deep recurrent neural network-based identification of precursor micrornas," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2895–2904, 2017.

[51] B. Lee, J. Baek, S. Park, and S. Yoon, "deeptarget: end-to-end learning framework for microrna target prediction using deep recurrent neural networks," in *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 434–442, 2016.

[52] J. J. A. Armenteros, C. K. Sønderby, S. K. Sønderby, H. Nielsen, and O. Winther, "Deeploc: prediction of protein subcellular localization using deep learning," *Bioinformatics*, vol. 33, no. 21, pp. 3387–3395, 2017.

[53] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[54] Y. Cao and Y. Shen, "Tale: Transformer-based protein function annotation with joint sequence-label embedding," *bioRxiv*, 2020.

[55] L. Chen, X. Tan, D. Wang, F. Zhong, X. Liu, T. Yang, X. Luo, K. Chen, H. Jiang, and M. Zheng, "Transformercpi: improving compound–protein interaction prediction by sequence-based deep learning with self-attention mechanism and label reversal experiments," *Bioinformatics*, vol. 36, no. 16, pp. 4406–4414, 2020.

[56] N. H. G. R. Institute, "Talking glossary of genetic terms," June 2021.

[57] M. AlQuraishi, "End-to-end differentiable learning of protein structure," *Cell systems*, vol. 8, no. 4, pp. 292–301, 2019.

[58] V. I. Jurtz, A. R. Johansen, M. Nielsen, J. J. Almagro Armenteros, H. Nielsen, C. K. Sønderby, O. Winther, and S. K. Sønderby, "An introduction to deep learning on biological sequence data: examples and solutions," *Bioinformatics*, vol. 33, no. 22, pp. 3685–3690, 2017.

[59] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, "Learned protein embeddings for machine learning," *Bioinformatics*, vol. 34, no. 15, pp. 2642–2648, 2018.

[60] W. Li, J. Köster, H. Xu, C.-H. Chen, T. Xiao, J. S. Liu, M. Brown, and X. S. Liu, "Quality control, modeling, and visualization of crispr screens with mageck-vispr," *Genome biology*, vol. 16, no. 1, pp. 1–13, 2015.

[61] C. M. Lee, T. H. Davis, and G. Bao, "Examination of crispr/cas9 design tools and the effect of target site accessibility on cas9 activity," *Experimental physiology*, vol. 103, no. 4, pp. 456–460, 2018.

[62] B. Zetsche, J. S. Gootenberg, O. O. Abudayyeh, I. M. Slaymaker, K. S. Makarova, P. Essletzbichler, S. E. Volz, J. Joung, J. Van Der Oost, A. Regev, *et al.*, "Cpf1 is a single rna-guided endonuclease of a class 2 crispr-cas system," *Cell*, vol. 163, no. 3, pp. 759–771, 2015.

[63] J. K. Hur, K. Kim, K. W. Been, G. Baek, S. Ye, J. W. Hur, S.-M. Ryu, Y. S. Lee, and J.-S. Kim, "Targeted mutagenesis in mice by electroporation of cpf1 ribonucleoproteins," *Nature biotechnology*, vol. 34, no. 8, pp. 807–808, 2016.

[64] H. Xu, T. Xiao, C.-H. Chen, W. Li, C. A. Meyer, Q. Wu, D. Wu, L. Cong, F. Zhang, J. S. Liu, *et al.*, "Sequence determinants of improved crispr sgrna design," *Genome research*, vol. 25, no. 8, pp. 1147–1157, 2015.

[65] B. Zetsche, M. Heidenreich, P. Mohanraju, I. Fedorova, J. Kneppers, E. M. DeGennaro, N. Winblad, S. R. Choudhury, O. O. Abudayyeh, J. S. Gootenberg, *et al.*, "Multiplex gene editing by crispr–cpf1 using a single crrna array," *Nature biotechnology*, vol. 35, no. 1, pp. 31–34, 2017.

[66] R. Xu, R. Qin, H. Li, D. Li, L. Li, P. Wei, and J. Yang, "Generation of targeted mutant rice using a crispr-cpf1 system," *Plant biotechnology journal*, vol. 15, no. 6, pp. 713–717, 2017.

[67] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a cpu and gpu math expression compiler," in *Proceedings of the Python for scientific computing conference (SciPy)*, vol. 4, pp. 1–7, Austin, TX, 2010.

[68] M. Haeussler, K. Schönig, H. Eckert, A. Eschstruth, J. Mianné, J.-B. Renaud, S. Schneider-Maunoury, A. Shkumatava, L. Teboul, J. Kent, *et al.*, "Evaluation of off-target and on-target scoring algorithms and integration into the guide rna selection tool crispor," *Genome biology*, vol. 17, no. 1, pp. 1–12, 2016.

[69] E. P. Consortium *et al.*, "An integrated encyclopedia of dna elements in the human genome," *Nature*, vol. 489, no. 7414, p. 57, 2012.

[70] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short dna sequences to the human genome," *Genome biology*, vol. 10, no. 3, pp. 1–10, 2009.

[71] B. P. Kleinstiver, S. Q. Tsai, M. S. Prew, N. T. Nguyen, M. M. Welch, J. M. Lopez, Z. R. McCaw, M. J. Aryee, and J. K. Joung, "Genome-wide specificities of crispr-cas cpf1 nucleases in human cells," *Nature biotechnology*, vol. 34, no. 8, pp. 869–874, 2016.

[72] R. Chari, N. C. Yeo, A. Chavez, and G. M. Church, "sgrna scorer 2.0: a species-independent model to predict crispr/cas9 activity," *ACS synthetic biology*, vol. 6, no. 5, pp. 902–904, 2017.

[73] D. Kim, J. Kim, J. K. Hur, K. W. Been, S.-h. Yoon, and J.-S. Kim, "Genome-wide analysis reveals specificities of cpf1 endonucleases in human cells," *Nature biotechnology*, vol. 34, no. 8, pp. 863–868, 2016.

[74] D. M. Garcia, D. Baek, C. Shin, G. W. Bell, A. Grimson, and D. P. Bartel, "Weak seed-pairing stability and high target-site abundance de-

crease the proficiency of lsy-6 and other micrornas," *Nature structural & molecular biology*, vol. 18, no. 10, p. 1139, 2011.

[75] A. Grimson, K. K.-H. Farh, W. K. Johnston, P. Garrett-Engele, L. P. Lim, and D. P. Bartel, "Microrna targeting specificity in mammals: determinants beyond seed pairing," *Molecular cell*, vol. 27, no. 1, pp. 91–105, 2007.

[76] D. Kim, Y. M. Sung, J. Park, S. Kim, J. Kim, J. Park, H. Ha, J. Y. Bae, S. Kim, and D. Baek, "General rules for functional microrna targeting," *Nature genetics*, vol. 48, no. 12, p. 1517, 2016.

[77] A. Krek, D. Grün, M. N. Poy, R. Wolf, L. Rosenberg, E. J. Epstein, P. MacMenamin, I. Da Piedade, K. C. Gunsalus, M. Stoffel, *et al.*, "Combinatorial microrna target predictions," *Nature genetics*, vol. 37, no. 5, pp. 495–500, 2005.

[78] J. P. Broughton, M. T. Lovci, J. L. Huang, G. W. Yeo, and A. E. Pasquinelli, "Pairing beyond the seed supports microrna targeting specificity," *Molecular cell*, vol. 64, no. 2, pp. 320–333, 2016.

[79] V. Agarwal, G. W. Bell, J.-W. Nam, and D. P. Bartel, "Predicting effective microrna target sites in mammalian mrnas," *elife*, vol. 4, p. e05005, 2015.

[80] A. Pla, X. Zhong, and S. Rayner, "miraw: A deep learning-based approach to predict microrna targets by analyzing whole microrna transcripts," *PLoS computational biology*, vol. 14, no. 7, p. e1006185, 2018.

[81] D. Betel, A. Koppal, P. Agius, C. Sander, and C. Leslie, "Comprehensive modeling of microrna targets predicts functional non-conserved and non-canonical sites," *Genome biology*, vol. 11, no. 8, p. R90, 2010.

[82] N. Wong and X. Wang, "mirdb: an online resource for microrna target prediction and functional annotations," *Nucleic acids research*, vol. 43, no. D1, pp. D146–D152, 2015.

[83] S. M. Peterson, J. A. Thompson, M. L. Ufkin, P. Sathyanarayana, L. Liaw, and C. B. Congdon, "Common features of microrna target prediction tools," *Frontiers in genetics*, vol. 5, p. 23, 2014.

[84] J. Sheu-Gruttadauria, Y. Xiao, L. F. Gebert, and I. J. MacRae, "Beyond the seed: structural basis for supplementary micro rna targeting by human argonaute2," *The EMBO journal*, vol. 38, no. 13, p. e101153, 2019.

[85] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[86] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[87] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch," *NIPS Autodiff Workshop*, 2017.

[88] I. S. Vlachos, M. D. Paraskevopoulou, D. Karagkouni, G. Georgakilas, T. Vergoulis, I. Kanellos, I.-L. Anastasopoulos, S. Maniou, K. Karathanou, D. Kalfakakou, *et al.*, "Diana-tarbase v7. 0: indexing more than half a million experimentally supported mirna: mrna interactions," *Nucleic acids research*, vol. 43, no. D1, pp. D153–D159, 2015.

[89] C.-H. Chou, N.-W. Chang, S. Shrestha, S.-D. Hsu, Y.-L. Lin, W.-H. Lee, C.-D. Yang, H.-C. Hong, T.-Y. Wei, S.-J. Tu, *et al.*, "mirtarbase

2016: updates to the experimentally validated mirna-target interactions database," *Nucleic acids research*, vol. 44, no. D1, pp. D239–D247, 2016.

[90] S. Grosswendt, A. Filipchyk, M. Manzano, F. Klironomos, M. Schilling, M. Herzog, E. Gottwein, and N. Rajewsky, "Unambiguous identification of mirna: target site interactions by different types of ligation reactions," *Molecular cell*, vol. 54, no. 6, pp. 1042–1054, 2014.

[91] A. Helwak, G. Kudla, T. Dudnakova, and D. Tollervey, "Mapping the human mirna interactome by clash reveals frequent noncanonical binding," *Cell*, vol. 153, no. 3, pp. 654–665, 2013.

[92] R. Lorenz, S. H. Bernhart, C. H. Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker, "Viennarna package 2.0," *Algorithms for molecular biology*, vol. 6, no. 1, p. 26, 2011.

[93] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[94] J. M. Berg, J. L. Tymoczko, and L. Stryer, "Biochemistry. 5th," *New York: WH Freeman*, vol. 38, no. 894, p. 76, 2006.

[95] S. R. Eddy, "Where did the blosum62 alignment score matrix come from?," *Nature biotechnology*, vol. 22, no. 8, pp. 1035–1036, 2004.

[96] J. Söding, A. Biegert, and A. N. Lupas, "The hhpred interactive server for protein homology detection and structure prediction," *Nucleic acids research*, vol. 33, no. suppl_2, pp. W244–W248, 2005.

[97] T. E. Creighton, *Proteins: structures and molecular properties*. Macmillan, 1993.

[98] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositional-

ity," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[99] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[100] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[101] M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, and B. Rost, "Modeling aspects of the language of life through transfer-learning protein sequences," *BMC bioinformatics*, vol. 20, no. 1, p. 723, 2019.

[102] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, D. Bhowmik, *et al.*, "Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing," *arXiv preprint arXiv:2007.06225*, 2020.

[103] A. Rives, S. Goyal, J. Meier, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *bioRxiv*, p. 622803, 2019.

[104] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[105] R. D. Finn, A. Bateman, J. Clements, P. Coggill, R. Y. Eberhardt, S. R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, *et al.*, "Pfam:

the protein families database," *Nucleic acids research*, vol. 42, no. D1, pp. D222–D230, 2014.

[106] T. L. Bailey, N. Williams, C. Misleh, and W. W. Li, "Meme: discovering and analyzing dna and protein sequence motifs," *Nucleic acids research*, vol. 34, no. suppl_2, pp. W369–W373, 2006.

[107] A. Elofsson and E. Sonnhammer, "A comparison of sequence and structure protein domain families as a basis for structural genomics.," *Bioinformatics (Oxford, England)*, vol. 15, no. 6, pp. 480–500, 1999.

[108] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, "Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures," *Nucleic acids research*, vol. 42, no. D1, pp. D304–D309, 2013.

[109] S. Khurana, R. Rawi, K. Kunji, G.-Y. Chuang, H. Bensmail, and R. Mall, "Deepsol: a deep learning framework for sequence-based protein solubility prediction," *Bioinformatics*, vol. 34, no. 15, pp. 2605–2613, 2018.

[110] G. J. Rocklin, T. M. Chidyausiku, I. Goreshnik, A. Ford, S. Houliston, A. Lemak, L. Carter, R. Ravichandran, V. K. Mulligan, A. Chevalier, *et al.*, "Global analysis of protein folding using massively parallel design, synthesis, and testing," *Science*, vol. 357, no. 6347, pp. 168–175, 2017.

[111] K. S. Sarkisyan, D. A. Bolotin, M. V. Meer, D. R. Usmanova, A. S. Mishin, G. V. Sharonov, D. N. Ivankov, N. G. Bozhanova, M. S. Baranov, O. Soylemez, *et al.*, "Local fitness landscape of the green fluorescent protein," *Nature*, vol. 533, no. 7603, pp. 397–401, 2016.

[112] M. S. Klausen, M. C. Jespersen, H. Nielsen, K. K. Jensen, V. I. Jurtz, C. K. Soenderby, M. O. A. Sommer, O. Winther, M. Nielsen, B. Petersen, *et al.*, "Netsurfp-2.0: Improved prediction of protein structural

features by integrated deep learning," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 6, pp. 520–527, 2019.

[113] K. D. Tsirigos, C. Peters, N. Shu, L. Käll, and A. Elofsson, "The topcons web server for consensus prediction of membrane protein topology and signal peptides," *Nucleic acids research*, vol. 43, no. W1, pp. W401–W407, 2015.

[114] L. S. Tavares, C. d. S. F. d. Silva, V. C. Souza, V. L. d. Silva, C. G. Diniz, and M. D. O. Santos, "Strategies and molecular tools to fight antimicrobial resistance: resistome, transcriptome, and antimicrobial peptides," *Frontiers in microbiology*, vol. 4, p. 412, 2013.

[115] J. H. Steiger, "Tests for comparing elements of a correlation matrix.," *Psychological bulletin*, vol. 87, no. 2, p. 245, 1980.

[116] K. Ratheesh, N. N. S., A. S. P., D. Sinha, V. B. Veedin Rajan, V. K. Esthaki, and P. D'Silva, "Hspir: a manually annotated heat shock protein information resource," *Bioinformatics*, vol. 28, no. 21, pp. 2853–2855, 2012.

[117] R. Kumar, B. Kumari, and M. Kumar, "Predhsp: sequence based proteome-wide heat shock protein prediction and classification tool to unlock the stress biology," *PloS one*, vol. 11, no. 5, p. e0155872, 2016.

[118] P. K. Meher, T. K. Sahu, S. Gahoi, and A. R. Rao, "ir-hsp: improved recognition of heat shock proteins, their families and sub-types based on g-spaced di-peptide features and support vector machine," *Frontiers in genetics*, vol. 8, p. 235, 2018.

[119] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, *et al.*, "The swiss-prot protein knowledgebase and its supplement trembl in 2003," *Nucleic acids research*, vol. 31, no. 1, pp. 365–370, 2003.

[120] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[121] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers," in *ACL*, 2019.

[122] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, p. ., 2017.

[123] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult, "Critical assessment of methods of protein structure prediction (casp)—round xiii," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 12, pp. 1011–1020, 2019.

[124] R. Poplin, P.-C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, J. Dijamco, N. Nguyen, P. T. Afshar, *et al.*, "A universal snp and small-indel variant caller using deep neural networks," *Nature biotechnology*, vol. 36, no. 10, pp. 983–987, 2018.

[125] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[126] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, "Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome," *bioRxiv*, 2020.

[127] R. Rao, J. Liu, R. Verkuil, J. Meier, J. F. Canny, P. Abbeel, T. Sercu, and A. Rives, "Msa transformer," *bioRxiv*, 2021.

[128] T. He, Z. Zhang, H. Zhang, *et al.*, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 558–567, 2019.

[129] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[130] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[131] E. Ben-Baruch, T. Ridnik, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "Asymmetric loss for multi-label classification," *arXiv preprint arXiv:2009.14119*, 2020.

[132] H.-S. Choi, D. Jung, S. Kim, and S. Yoon, "Imbalanced data classification via cooperative interaction between classifier and generator," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

초 록

우리는 빅데이터의 시대를 맞이하고 있으며, 의생명 분야 또한 예외가 아니다. 차세대 염기서열 분석과 같은 기술들이 도래함에 따라, 폭발적인 의생명 데이터의 증가를 활용하기 위한 방법론의 개발은 생물정보학 분야의 주요 과제 중의 하나이다. 심층 학습을 포함한 표현 학습 기법들은 인공지능 학계가 오랫동안 어려움을 겪어온 다양한 분야에서 상당한 발전을 이루었다. 표현 학습은 생물정보학 분야에서도 많은 가능성을 보여주었다. 하지만 단순한 적용으로는 생물학적 서열 데이터 분석의 성공적인 결과를 항상 얻을 수는 않으며, 여전히 연구가 필요한 많은 문제들이 남아있다.

본 학위논문은 생물학적 서열 데이터 분석과 관련된 세 가지 사안을 해결하기 위해, 표현 학습에 기반한 일련의 방법론들을 제안한다. 첫 번째로, 유전자가위 실험 데이터에 내재된 정보와 수율의 균형에 대처할 수 있는 2단계 학습 기법을 제안한다. 두 번째로, 두 염기 서열 간의 상호 작용을 학습하기 위한 부호화 방식을 제안한다. 세 번째로, 기하급수적으로 증가하는 특징되지 않은 단백질 서열을 활용하기 위한 자기 지도 사전 학습 기법을 제안한다. 요약하자면, 본 학위논문은 생물학적 서열 데이터를 분석하여 중요한 정보를 도출할 수 있는 표현 학습에 기반한 일련의 방법론들을 제안한다.

주요어: machine learning, deep learning, representation learning, artificial intelligence, biological sequence, CRISPR, microRNA target, protein

학번: 2015-20914