Neural Embeddings for kNN Search in Biological Sequence

Zhihao Chang[1], **Linzhu Yu**[2], **Yanchao Xu**[2], **Wentao Hu**[3]

[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
[2]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[3]Zhejiang Police College, Hangzhou, China
{changzhihao, linzhu, xuyanchao, wthu}@zju.edu.cn

Abstract

Biological sequence nearest neighbor search plays a fundamental role in bioinformatics. To alleviate the pain of quadratic complexity for conventional distance computation, neural distance embeddings, which project sequences into geometric space, have been recognized as a promising paradigm. To maintain the distance order between sequences, these models all deploy triplet loss and use intuitive methods to select a subset of triplets for training from a vast selection space. However, we observed that such training often enables models to distinguish only a fraction of distance orders, leaving others unrecognized. Moreover, naively selecting more triplets for training under the state-of-the-art network not only adds costs but also hampers model performance.

In this paper, we introduce Bio-kNN: a kNN search framework for biological sequences. It includes a systematic triplet selection method and a multi-head network, enhancing the discernment of all distance orders without increasing training expenses. Initially, we propose a clustering-based approach to partition all triplets into several clusters with similar properties, and then select triplets from these clusters using an innovative strategy. Meanwhile, we noticed that simultaneously training different types of triplets in the same network cannot achieve the expected performance, thus we propose a multi-head network to tackle this. Our network employs a convolutional neural network (CNN) to extract local features shared by all clusters, and then learns a multi-layer perception (MLP) head for each cluster separately. Besides, we treat CNN as a special head, thereby integrating crucial local features which are neglected in previous models into our model for similarity recognition. Extensive experiments show that our Bio-kNN significantly outperforms the state-of-the-art methods on two large-scale datasets without increasing the training cost.

Introduction

Biological sequence nearest neighbor search plays a fundamental role in bioinformatics research and serves as the cornerstone for numerous tasks, including gene prediction (Chothia and Lesk 1986), homology analysis (Sander and Schneider 1991), sequence clustering (Steinegger and Söding 2018; Li and Godzik 2021), etc. Traditional methods for measuring global or local similarity between sequences rely on alignment based on dynamic programming. In this paper, we focus on the global similarity between sequences, evaluated by the widely used Needleman-Wunsch (NW) algorithm (Needleman and Wunsch 1970). While the NW algorithm is proficient in calculating sequence similarity with precision, its inherent quadratic complexity poses significant challenges for rapid analysis, particularly when dealing with large-scale datasets comprising sequences that extend to hundreds or even thousands of amino acids or nucleotides.

In recent years, embedding-based approaches have emerged as a promising paradigm for expediting sequence similarity analysis. These approaches involve projecting sequences into a geometric embedding space through an embedding function, such that the distance between sequences can be approximated by the distance in the embedding space, which offers a computationally efficient alternative. These approaches can be broadly divided into two categories based on the core idea of the embedding function: rule-based and neural network-based. Rule-based approaches (Sims et al. 2009; Gao and Qi 2007; Ulitsky et al. 2006; Haubold et al. 2009; Leimeister and Morgenstern 2014) often rely on some predefined encoding rules. Several studies (Corso et al. 2021; Chen et al. 2022) have indicated that, in multiple tasks, these approaches exhibit inferior performance compared to neural network-based ones. Given this context, we will not delve into rule-based approaches, and instead concentrate on exploring neural network-based approaches.

Existing research on neural network-based methods (Zheng et al. 2019; Chen et al. 2022; Zhang, Yuan, and Indyk 2019; Dai et al. 2020; Corso et al. 2021) primarily focused on various components such as encoding models and loss functions. These components are tailored to the task for which the learned embeddings are used. Notably, certain approaches (Zhang, Yuan, and Indyk 2019; Dai et al. 2020) focus on the learning objective aimed at preserving distance orders within the embedding space to facilitate kNN searches. To achieve this goal, these approaches employ triplet loss (Weinberger and Saul 2009; Hermans, Beyer, and Leibe 2017) and use intuitive methods to select triplets in the form $(S_{acr}, S_{pos}, S_{neg})$ for training, in which S_{acr} is the anchor sequence, S_{pos} is the positive sequence that has smaller distance to S_{acr} than the negative sequence S_{neg}. However, we found that the models trained by these methods exhibit proficiency in distance order recognition

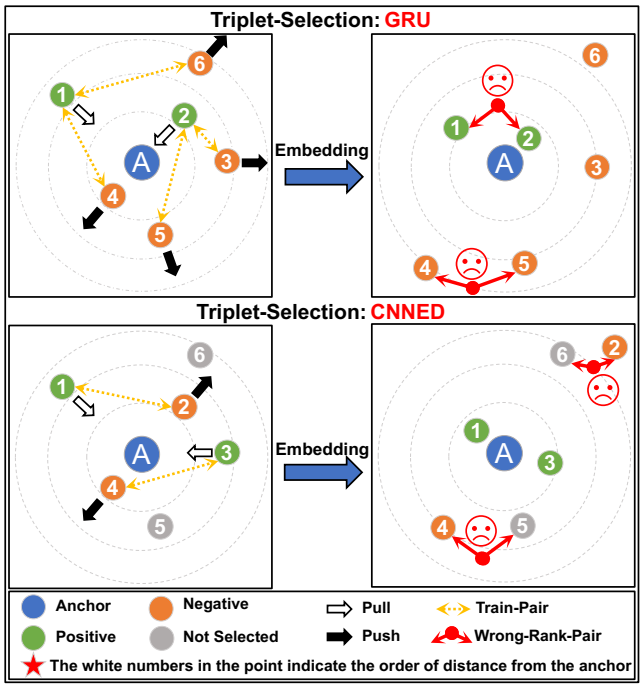

Figure 1: The triplet selection methods used by GRU (Zhang, Yuan, and Indyk 2019) and CNNED (Dai et al. 2020). For GRU, the Top-N closest to the anchor is positive and the others are negative; for CNNED, two sequences are randomly selected from the Top-K closest to the anchor, the closer is positive, and the farther is negative. In this example we set N equal to 2 and K equal to 4.

for only a limited subset of triplets, rather than the entire set. As illustrated in Figure 1, while certain order relations may be accurately identified after encoding, overlooked relations during training can substantially compromise the results. Such complications stem from that each sequence lacks a definitive category label, rendering existing techniques ineffective in this context. It might be hypothesized that increasing the number of triplets for training could ameliorate this issue. However, our assessments within a state-of-the-art network indicate that such problem has not been alleviated while suffering additional training expanses.

In this paper, we introduce Bio-kNN, a biological sequence kNN search framework. Bio-kNN aims to notably improve the recognition accuracy of distance order distributed throughout the whole space without augmenting training expenses. The core idea of Bio-kNN is to partition all triplets into several clusters based on certain properties and learn a feature extraction network for each cluster. Specifically, Bio-kNN features two main modules: (1)**Triplet selection method.** A notable limitation of previous models is that only a subset of the triplets is considered during training. In this module, we consider all possible combinations of triplets. We partition the selection space into small cells and merge cells with similar distance distributions into several clusters. We then employ an innovative strategy to select training triplets from these clusters without external samples. (2)**Multi-head network.** We noticed that merely adding more triplets in the SOTA network does not improve the performance, we thus propose a multi-head network to address it. Our network uses CNN as the backbone to extract local features, and learns a multi-layer perception head for each cluster to extract global features. Furthermore, we integrate previously overlooked local features derived from the CNN, which are crucial in discernment.

To summarize, we made four contributions in this paper.

1. We consider the entire selection space instead of subsets, and propose a clustering-based triplet selection method.
2. We notice that the performance of SOTA network degrades when simultaneously training different types of triplets. A multi-head network is designed to alleviate it.
3. We treat CNN as a special head and integrate crucial local features into our model for sequence similarity.
4. We conduct extensive experiments on two large-scale datasets, and the results show that our method significantly outperforms the state-of-the-art methods.

Related Work

Rule-Based Approaches. Numerous rule-based approaches have been proposed over the past few decades, which can be broadly classified into two categories. The first category typically utilizes word frequency statistics with a predefined length (Kariin and Burge 1995) or the information content of word frequency distribution (Sims et al. 2009; Gao and Qi 2007) as features to characterize sequence similarity. On the other hand, the second category of methods is based on the concept of sub-strings (Ulitsky et al. 2006; Haubold et al. 2009; Leimeister and Morgenstern 2014). However, it should be noted that all these approaches are data-independent, and their distance measures rely on heuristic rules. Several studies have shown that these approaches exhibit weaker performance compared to neural network-based approaches across various tasks.

Neural Network-Based Approaches. Notable efforts in neural networks have been made to approximate distances for biological sequences in recent years. SENSE (Zheng et al. 2019) is the first attempt to employ neural networks for comparison-free sequence analysis by utilizing a convolutional neural network. However, SENSE is restricted to handling sequences of the same length. To address it, AS-Mac (Chen et al. 2022) was proposed, which employs an approximate string matching algorithm to extract relevant features through neural network. Regrettably, the performance of this approach degrades when dealing with protein sequences, primarily due to the massive search space involved.

A research domain closely aligned with our work focuses on edit distance embedding. The distinction lies in the NW algorithm's requirement to normalize the edit distance by a dynamically varying length, thereby amplifying the complexity of discerning similarities. CGK (Ostrovsky and Rabani 2007) embeds the edit distance into the hamming space with a distortion of $2^{O(\sqrt{\log l \log\log l})}$, however, this algorithm is excessively intricate for practical application. Zhang et al. (Zhang, Yuan, and Indyk 2019) propose a

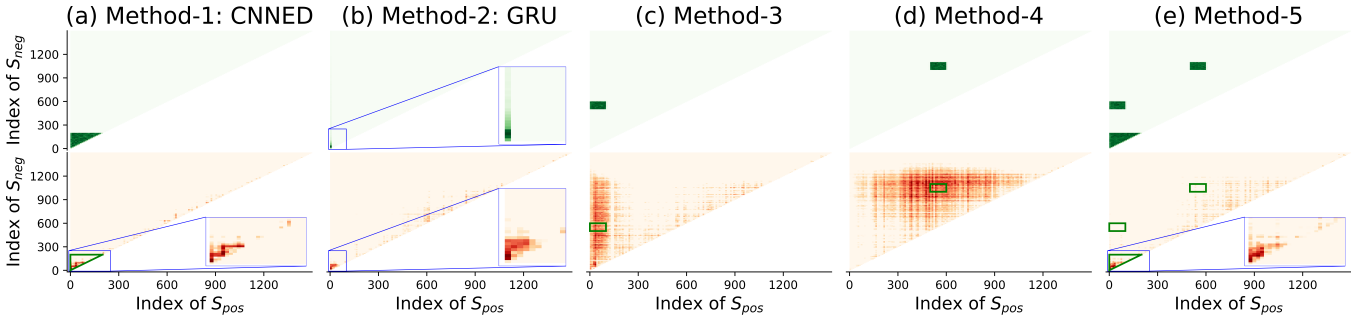

Figure 2: Motivating example. For the convenience of observation, the bottom subfigures are the results after comparing with the model trained by randomly selecting triplets, i.e., for each S_{acr}, two sequences are randomly selected from the training set, the closer to S_{acr} is S_{pos}, and the farther is S_{neg}.

two-layer GRU structure to encode sequences, dividing the training process into three stages and utilizing three different loss functions. Nonetheless, the embedding dimension generated by this method is relatively high, resulting in substantial memory consumption. CNNED (Dai et al. 2020) discovers that an untrained random CNN performs comparable to GRU models, leading to the belief that the CNN is more suitable for the edit distance embedding than RNN-based models. NeuroSEED (Corso et al. 2021) explores the potential of employing global and local transformers to encode biological sequences, and experimental results also affirm that convolutional models surpass feedforward and recurrent models for biological sequence edit distance tasks. Furthermore, NeuroSEED proposes that the hyperbolic space can better capture the data dependencies among biological sequences from the perspective of embedded geometry.

Motivating Example

In this section, we use an example to reveal the limitations of existing methods. We first model the entire selection space as an upper triangular area. Then we visualize the distribution of training triplets and the performance of the trained model, thus we can easily observe the relationship between them. Example details are as follows.

Example Setting

We first randomly select 3000 sequences from UniProtKB[1] and use 1500 of them as the training set, while the remaining 1500 as the test set. Then, we employ the state-of-the-art pipeline proposed by CNNED (Dai et al. 2020) as the common training framework, and replace the triplet selection method with five other methods respectively during training, including two methods adopted by previous models: the methods used by CNNED (Dai et al. 2020) and GRU (Zhang, Yuan, and Indyk 2019), and three methods designed for comparison: Method-3, Method-4, and Method-5.

In Figure 2, we plot the distribution of triplets selected by these five methods on the **training set** (top subfigures) and the distance order recognition results on the **test set** (bottom subfigures) respectively. The horizontal and vertical coordinates (i, j) of each subfigure in Figure 2 are all determined by the triplet $(S_{acr}, S_{pos}, S_{neg})$. For each S_{acr}, we first sort other sequences according to the distance between them and S_{acr} from small to large to form a list, and the indices i and j of S_{pos} and S_{neg} in the list are used as the abscissa and ordinate, respectively. The difference between the top and the bottom subfigures is the triplets used for visualization: (1) We plot top subfigures according to the triplets obtained in the **training set** by the five triplet selection methods. The depth of the color indicates the frequency of the corresponding triplet is selected. (2) For the bottom subfigures, the triplets are all triplets combinations in the **test set**, and these subfigures are used to visualize the results of distance order recognition in the test set. We iterated all triplet combinations in the test set to check whether the distance between S_{acr} and S_{pos} is smaller than the distance between S_{acr} and S_{neg} after encoding by model f, i.e., $dist_e(f(S_{acr}), f(S_{pos})) < dist_e(f(S_{acr}), f(S_{neg}))$, the more frequency of the match, the more vivid the color.

Phenomenon

From Figure 2, we can observe three following phenomena, including one expected and two inconsistent with the expectation but interesting:

1. **Expected.** Figure 2(a)-(d) illustrate that sequence distance order recognition in the test set is highly correlated with the training triplets. This phenomenon is expected, as the more triplets the model learns for a region in the training set, the better it helps distinguish the order of that region in the test set. *However, we can clearly observe that the model trained by these methods can only recognize the order of a small part of the whole area.* This observation shows that the model is very limited in identifying crucial regions that lie beyond its training region(e.g., let the model in Figure 2(a) recognize the order of the region determined by Method-3). Such limitation greatly affects the effectiveness of the model.
2. **UnExpected.** Inspired by phenomenon 1, an intuitive idea is to select more training regions. We thus trained Method-5, which simultaneously trains the regions selected by Method-1, Method-3, and Method-4. However, the recognition results are not consistent with our expectation, as shown in the Figure 2(e), although certain re-

[1]https://www.uniprot.org/

gions have been trained, the corresponding regions in the test set do not have better distance order recognition.

3. **UnExpected.** These figures also illustrate that the model has a radiation effect on regions outside the training region, i.e. even if some regions are not selected, the model is also better able to recognize the order of this region. Furthermore, the radiation region produced by the training region in different positions varies greatly.

Method

To address the issues arising in phenomenon 2, we propose Bio-kNN, which includes a triplet selection method and a multi-head network. Its framework is shown in Figure 3.

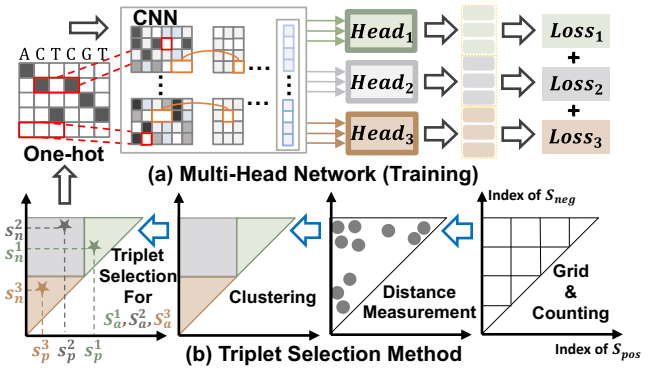

Figure 3: The Framework of Bio-kNN.

Triplet Selection Method

Partition Selection Space. As shown in Figure 3(b), we partition the entire selection space formed by the training set into small cells. Specifically, we use the same setting as that in the motivating example to model the entire selection space as an upper triangular area, and the length of two legs of the triangle is the number of sequences in the training set. In this setting, for each S_{acr}, each point in the triangle represents a triplet, where the abscissa represents the index of the S_{pos}, and the ordinate represents the index of the S_{neg}. Then, we divide the horizontal and vertical axes into B groups respectively based on an equal interval δ, where the horizontal axis is divided into $[[x_0, x_1), [x_1, x_2), ..., [x_{B-1}, x_B)]$ and the vertical is $[[y_0, y_1), [y_1, y_2), ..., [y_{B-1}, y_B)]$. Thus the upper triangular area is divided into $\sum_{i=1}^{B} i$ small cells, where most of the cells are grids and few are triangles. Then, the coordinates of each cell can be described as (X_i, Y_j), where X_i means $[x_i, x_{i+1})$, and Y_j means $[y_j, y_{j+1})$.

Distribution Statistics in Cell. For each cell after partitioning, we use the interval of the coordinates to count the horizontal and vertical distributions. This step is inspired by phenomenon 3 in the motivating example, which shows that some properties between adjacent regions may be similar. In this step, we try to use the intuitive distance distribution as this property. It is worth noting that the possibility of other properties is not ruled out, which can be studied in the future. Next, we use an example to illustrate the details of our approach. Suppose there is a cell with coordinates $(X_{500,600}, Y_{700,800})$, we use each sequence in the training set as S_{acr} in turn. For each S_{acr}, we sort other sequences in the training set according to the NW distance between them and S_{acr} from small to large to form a list l. We then count the horizontal distance distribution between all sequences in the list $l[500 : 600]$ and S_{acr} for $X_{500,600}$, while count $l[700 : 800]$ for $Y_{700,800}$. In this way, the coordinates of each cell can be further described as (X_i, Y_j), where X_i means $count([x_i, x_{i+1}))$, and Y_j means $count([y_j, y_{j+1}))$. Subsequent cell coordinates will use this definition by default.

Distance Measurement between Cells. How to measure the distance between cells with distributions as coordinates becomes a new problem. Currently, there are many functions to measure the distance between two distributions, such as Kullback–Leibler divergence (Kullback and Leibler 1951), Jensen-Shannon divergence (Fuglede and Topsøe 2004), Earth mover's distance (EMD) (Rubner, Tomasi, and Guibas 2000), etc. However, we noticed that when two distributions do not overlap, the KL divergence is meaningless, and the JS divergence is a constant, so neither of these functions is suitable for measuring the distance between cells in our application scenario. Considering that EMD as a metric satisfies non-negativity, symmetry, and triangle inequality, we define the distance between two cells on the basis of EMD. Specifically, given any two cells p and q, their coordinates are (X_{p_i}, Y_{p_j}) and (X_{q_i}, Y_{q_j}) respectively, then we define the distance $d_{cell}(p, q)$ between p and q is:

$$d_{cell}(p, q) = EMD(X_{p_i}, X_{q_i}) + EMD(Y_{p_j}, Y_{q_j}) \quad (1)$$

We prove that $d_{cell}(p, q)$ between cells is still a metric.

Theorem 1 *The distance d_{cell} computed by Equation 1 is a metric. Given three any cells p, q, and r, we have:*
(1) Non-negativity. *If p != q, then $d_{cell}(p, q) > 0$.*
(2) Symmetry. *$d_{cell}(p, q) = d_{cell}(p, q)$.*
(3) Triangle inequality. *$d_{cell}(p, r) \leq d_{cell}(p, q) + d_{cell}(q, r)$*

Proof 1 *According to the non-negativity and symmetry of the EMD, it can be easily obtained that d_{cell} also satisfies the non-negativity and symmetry, so we will only prove the triangle inequality of d_{cell}.*

$$\begin{aligned} d_{cell}(p, r) &= EMD(X_{p_i}, X_{r_i}) + EMD(Y_{p_j}, Y_{r_j}) \\ &\leq (EMD(X_{p_i}, X_{q_i}) + EMD(X_{q_i}, X_{r_i})) \\ &+ (EMD(Y_{p_j}, Y_{q_j}) + EMD(Y_{q_j}, Y_{r_j})) \\ &= (EMD(X_{p_i}, X_{q_i}) + EMD(Y_{p_j}, Y_{q_j}) \\ &+ (EMD(X_{q_i}, X_{r_i}) + EMD(Y_{q_j}, Y_{r_j})) \\ &= d_{cell}(p, q) + d_{cell}(q, r) \end{aligned}$$

Cell Clustering. Our last step is to merge those cells that have a similar distance distribution. We achieve this using unsupervised clustering, which is naturally suited to distinguishing similar items such that distributions vary widely across clusters, while the distribution of cells within a single cluster is very close. In this paper, we do not propose a new clustering algorithm, but directly deploy existing clustering algorithms. In following, we evaluated the performance of commonly used clustering algorithms such as k-means

(Forgy 1965), agglomerative(Murtagh and Contreras 2012), and spectral clustering(von Luxburg 2007). Subsequent experiments will show more detailed results.

Selection Strategy. Suppose there are m training sequences and n clusters are obtained based on the above method. An intuitive selection strategy is that for each S_{acr}, We randomly select one point from each of the n clusters at each epoch, and the abscissa of these n points is the index of the S_{pos}, the ordinate is the index of the S_{neg}. However, this strategy needs to select $m*n$ triplets for training at each epoch. Clearly, the training cost of this strategy increases linearly with m and n, and it will bring burden to the expansion of the dataset when n have a large value.

We employ a novel selection strategy that can achieve good performance without adding more cost. Specifically, before each epoch of training, we first randomly shuffle all anchor sequences. Then for each batch, we divide all anchor sequences in the current batch evenly into n lists, and assign the n clusters to the n lists as candidate clusters respectively. The strategy at this time is that for each S_{acr}, we only randomly select a point from its corresponding candidate clusters instead of all clusters, and the number of training triplets for each epoch is also changed from $m * n$ to n.

Multi-Head Network

Network Structure. In recent years, several works (Dai et al. 2020; Corso et al. 2021) have shown that convolutional models outperform feedforward and recurrent models for sequence embedding, so our learning model utilizes the CNN submodule in CNNED (Dai et al. 2020) as a general backbone. Subsequently, multiple multi-layer perceptron (MLP) heads are deployed in parallel following the convolutional layers, thereby facilitating the fusion of local features from different perspectives to extract global features. In this structure, the number of heads is the same as the number of candidate clusters k. Each head has exactly the same structure and is trained in parallel without communicating with each other. The core idea of our multi-head model is that we hope to learn one head for each candidate cluster, thus avoiding potential contradictions between candidate selection clusters during training. It is imperative to highlight that our model exhibits an obvious distinction between the training and inference phases, we introduce them separately below.

Training Phase. During the training phase as shown in Figure 3(a), we first use the selection method introduced in the previous section to select a triplet $(S^i_{acr}, S^i_{pos}, S^i_{neg})$ for each anchor sequence in a batch. Then, the one-hot embedding representations $(\boldsymbol{X^i_{acr}}, \boldsymbol{X^i_{pos}}, \boldsymbol{X^i_{neg}})$ of all these triplets are simultaneously fed into the CNN, which is encoded as $(\boldsymbol{y^i_{acr}}, \boldsymbol{y^i_{pos}}, \boldsymbol{y^i_{neg}})$. After CNN encoding, the flow of these triplets starts to fork, and triplets selected from different clusters are fed to different MLP heads. Specifically, The embedding function of our multi-head network during the training phase can be expressed as follows:

$$\boldsymbol{y^i_{acr}}, \boldsymbol{y^i_{pos}}, \boldsymbol{y^i_{neg}} = CNN(\boldsymbol{X^i_{acr}}, \boldsymbol{X^i_{pos}}, \boldsymbol{X^i_{neg}})$$
$$\boldsymbol{z^i_{acr}}, \boldsymbol{z^i_{pos}}, \boldsymbol{z^i_{neg}} = MLP^i(\boldsymbol{y^i_{acr}}, \boldsymbol{y^i_{pos}}, \boldsymbol{y^i_{neg}})$$

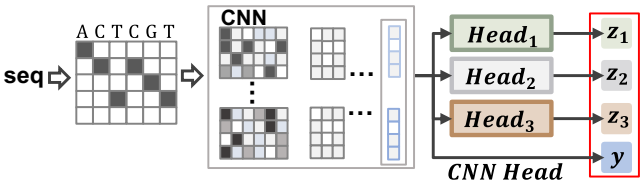

Figure 4: Multi-head Network (Inference).

after all triplets are encoded by the model, the final $loss$ is:

$$loss = \sum_{i=1}^{k} Loss(\boldsymbol{z^i_{acr}}, \boldsymbol{z^i_{pos}}, \boldsymbol{z^i_{neg}}) \tag{2}$$

where k represents both the number of candidate selection clusters and the number of heads, and $Loss$ is the combination of triplet loss and MSE loss.

Inference Phase. As depicted in Figure 4, we feed all sequences into the trained neutral network one by one during the inference phase. For each sequence, we use the one-hot representation $\boldsymbol{X}$ of this sequence and encode it through CNN, then feed the feature $\boldsymbol{y}$ output by CNN to all the MLP heads simultaneously. The outputs $[\boldsymbol{z^1}, ..., \boldsymbol{z^k}]$ of these heads are then all cascaded together. In addition, we treat CNN as a special head and concatenate the feature $\boldsymbol{y}$ output by CNN to the end. We will explain the reason for cascading CNN features below. The embedding function during the inference phase can be expressed as follows:

$$\boldsymbol{y} = CNN(\boldsymbol{X}) \tag{3}$$
$$\boldsymbol{z^i} = MLP^i(\boldsymbol{y}) \tag{4}$$

the representation of the sequence in embedding space is:

$$Embedding = [\boldsymbol{z^1}, ..., \boldsymbol{z^k}, \boldsymbol{y}] \tag{5}$$

CNN Serves as a Special Head. The core idea of our network is to train distinct MLP heads for each candidate cluster. Each of these heads aims to learn unique weights for the local features extracted by the CNN, essentially learning the most discriminative features that can distinguish different sequences within each cluster. However, fine-grained details can easily be ignored during learning. To alleviate the potential impact of these fine-grained feature losses, we introduce a compensation measure using CNN as a special head in the inference stage. Specifically, we concatenate local features with the final embedding, which is similar to the effect of fully connected layers with identity matrix and frozen weights. This approach effectively counteracts the adverse consequences of fine-grained features being ignored.

Embedding Geometry. There are many studies using various functions to calculate the distance between two embedding vectors, including Euclidean distance (Dai et al. 2020), Jaccard distance (Zheng et al. 2019), Hyperbolic distance (Corso et al. 2021), etc. However, for the multi-head network we designed, the final embedding of the sequence is the concatenation of vectors output by multiple heads. In order to make the features of each head play a bigger role in the distance calculation, we use a new metric instead of

directly using the Euclidean distance to calculate the distance between vectors. Specifically, we first calculate the Euclidean distance between the vectors output by a single head, and then sum the Euclidean distances of multiple heads as the final distance. Suppose there are two embedding vectors $\boldsymbol{x} = [\boldsymbol{x^1}, ..., \boldsymbol{x^k}, \boldsymbol{x^{cnn}}]$ and $\boldsymbol{y} = [\boldsymbol{y^1}, ..., \boldsymbol{y^k}, \boldsymbol{y^{cnn}}]$, then the distance between them can be described as:

$$dist_e(\boldsymbol{x}, \boldsymbol{y}) = Euc(\boldsymbol{x^{cnn}}, \boldsymbol{y^{cnn}}) + \sum_{i=1}^{k} Euc(\boldsymbol{x^i}, \boldsymbol{y^i}) \quad (6)$$

Experiments

Experimental Settings

Datasets. We evaluate our neural embeddings through the utilization of two extensively recognized datasets(Dai et al. 2020; Zhang, Yuan, and Indyk 2019), i.e., the **Uniprot** and **Uniref**. These datasets exhibit varying sizes and sequence lengths, and their properties are shown in the Table 1. Consistent with existing works, we partition each dataset into distinct subsets, namely the training set, query set, and base set. Both the training set and the query set are composed of 1,000 sequences, and the other items belong to the base set.

Dataset	Uniprot	Uniref
Alphabet Size	25	24
# Items	474741	395869
Avg-Length	376.47	442.84
Min-Length	2	201
Max-Length	4998	4998

Table 1: Dataset Statistics

Metrics. We follow existing works (Zhang, Yuan, and Indyk 2019; Dai et al. 2020) and use the task of nearest neighbor search to evaluate the effectiveness of our model, i.e. whether the distance order in the embedding space still preserves. Specifically, we use: (1) Top-k hitting ratio (HR@k). This metric is used to detect the overlap percentage of the top-k results and the ground truth. (2) Top-1 Recall. This one evaluates the performance of finding the most similar sequence to the query sequence by different methods.

Baselines. We adopt previous network-based approaches as baselines, including GRU (Zhang, Yuan, and Indyk 2019), CNNED (Dai et al. 2020), NeuroSEED (Corso et al. 2021), AsMac (Chen et al. 2022), where NeuroSEED can be further divided into Global (Global T.) and Local Transformer (Local T.). Since SENSE (Zheng et al. 2019) cannot be used for unequal-length datasets, and its performance has been proven to be weaker than AsMac, we will not use it as a baseline. To demonstrate the effectiveness of the selection method and multi-head network, we use **Bio-kNN-Base** to denote the method without cascading CNN features, and refer to the complete method as **Bio-kNN**.

Implementation Details. We use the EMBOSS[1] to compute the NW distance between sequences. In our implementation, we set the split interval $\delta = 100$ and experimentally tested the effect of various clustering algorithms and the number of clusters. Besides, we directly used the CNN submodule in CNNED. Code and datasets are available at https://github.com/Proudc/Bio-KNN.

[1]https://www.ebi.ac.uk/Tools/emboss/

Experimental results

Clustering-Based Triplet Selection. Tables 2 and 3 show the performance of Bio-kNN-Base under various clustering algorithms and the number of clusters, including k-means, agglomerative (HAC), spectral clustering, and non-clustering. These results show that:(1) With a fixed output dimension (128), the performance of Bio-kNN-Base consistently surpasses the non-clustering counterpart in various algorithms and the number of clusters, reaffirming the indispensability of segmenting the selection space. (2) HAC shows superior performance within certain configurations in contrast to the other two methods. This may be attributed to the ability of HAC to handle outlier cells more efficiently relative to other techniques, which also prompted us to use the HAC by default in subsequent experiments.

# Clusters*(D/h)	Method	HR@1	HR@10	HR@50
$1 * 128$	None	48.30	35.48	24.21
$2 * 64$	K-Means	48.60	36.51	25.19
$2 * 64$	HAC	48.60	36.51	25.19
$2 * 64$	Spectral	48.60	36.51	25.19
$4 * 32$	K-Means	49.90	38.58	26.98
4 * 32	**HAC**	**50.50**	**39.13**	**27.28**
$4 * 32$	Spectral	49.00	36.60	25.23
$8 * 16$	K-Means	49.70	37.90	26.00
$8 * 16$	HAC	48.80	37.52	25.70
$8 * 16$	Spectral	48.30	36.23	24.86

Note: D/h indicates the output dimension of each head.

Table 2: Uniprot: various clustering methods and # clusters

# Clusters*(D/h)	Method	HR@1	HR@10	HR@50
$1 * 128$	None	28.30	24.39	15.60
$2 * 64$	K-Means	31.10	26.91	17.54
2 * 64	**HAC**	**33.90**	**29.88**	**19.58**
$2 * 64$	Spectral	29.30	25.88	16.84
$4 * 32$	K-Means	30.70	25.83	16.63
$4 * 32$	HAC	32.40	26.92	17.42
$4 * 32$	Spectral	30.00	25.93	16.91
$8 * 16$	K-Means	31.70	26.80	17.41
$8 * 16$	HAC	32.20	26.57	17.34
$8 * 16$	Spectral	31.20	25.67	16.90

Table 3: Uniref: various clustering methods and # clusters

Embedding Effectiveness. Table 4 presents an overview of the performance exhibited by different methods concerning the top-k similarity search task. As shown, on both datasets, our method Bio-kNN significantly outperforms all methods on all metrics. Using the Uniprot dataset as an example, Bio-kNN yields a remarkable enhancement across

Model	Uniprot				Uniref			
	HR@1	HR@5	HR@10	HR@50	HR@1	HR@5	HR@10	HR@50
AsMac	47.07	32.60	24.25	9.93	20.57	11.93	8.08	2.68
GRU	40.83	40.05	34.53	23.16	30.73	26.53	22.73	13.62
CNNED	47.70	40.43	34.58	23.37	35.13	32.51	28.55	18.72
Global T.	48.76	39.97	34.16	22.29	27.80	22.38	18.67	10.47
Local T.	49.10	40.11	34.27	22.43	27.07	21.23	17.94	10.20
Bio-kNN	**54.00**	**48.31**	**42.69**	**30.28**	**37.60**	**36.18**	**32.51**	**21.13**
Gap With SOTA	+4.90	+7.88	+8.11	+6.91	+2.47	+3.67	+3.96	+2.41

Table 4: Embedding Results (repeat three times and report average results)

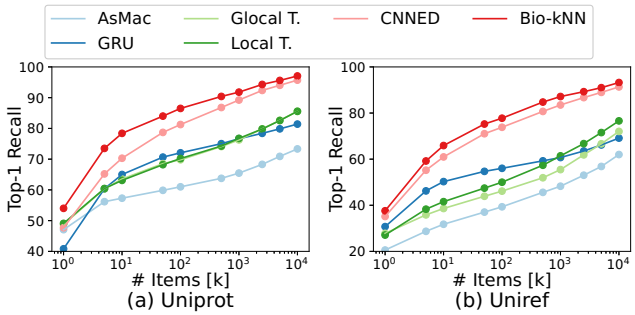

Figure 5: Top-1 Recall curves for multiple methods.

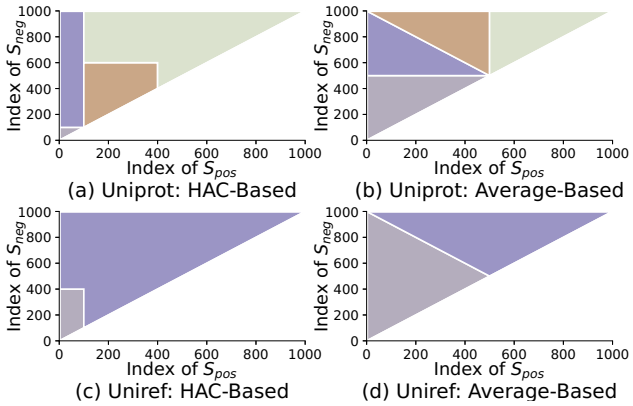

Figure 6: Segmentation Results of HAC(H) and Average(A)

metrics, ranging from 4.9% to 8.11% when compared to the state-of-the-art counterparts. Notably, a substantial majority of metrics experience an augmentation of over 6%. This non-negligible improvement is impressive given the fact that, unlike previous methods that only focus on partial subsets of triplets, Bio-kNN essentially partitions the entire selection space and learns individual heads for each distinct subspace. Besides, Bio-kNN incorporates the fine-grained local features extracted by CNN, which further improves its ability to distinguish similarities between sequences. We plot the curves of Top-1 recall for various methods on different datasets in Figure 5. We observe that our model also achieves significant performance gains on the task of finding the most similar sequence compared to other methods.

Ablation Studies. Our Bio-kNN comprises three modules: clustering-based triplet selection, a multi-head network, and CNN features. We conduct the following experiments to validate the contributions of these modules: (1) Considering that the necessity of segmenting the space has been verified in Table 2 and 3, we exclusively explore specific segmentation methods. We thus independently evaluate the segmentation outcomes on both sides of Figure 6. (2) Replacing the multi-head (M) network with a single-head (S) network. (3) Omitting the features extracted by CNN.

The results in Table 5 demonstrate that neglecting any of the three modules leads to a reduction in performance. The reason is that we take into account the distance distribution among cells when segmenting the selection space.

Datasets	Method	HR@1	HR@10	HR@50
Uniprot	H + S + CNN	53.20	41.02	28.56
	A + M + CNN	52.03	40.31	27.93
	H + M	50.40	39.01	27.26
	H + M + CNN	**54.00**	**42.69**	**30.28**
Uniref	H + S + CNN	35.63	30.31	19.75
	A + M + CNN	35.43	30.19	19.60
	H + M	33.67	28.95	18.86
	H + M + CNN	**37.60**	**32.51**	**21.13**

Table 5: Ablation Studies Results

Separate heads are assembled for clusters with large differences in distribution, making training more targeted. The fine-grained features extracted by CNN also effectively enhance the model's ability to distinguish sequence similarity.

Conclusion

We propose Bio-kNN for biological nearest neighbor search, which includes a clustering-based triplet selection method and a CNN-based multi-head network. It also incorporates local features extracted by CNN. Experimental results show that Bio-kNN outperforms the state-of-the-art.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities(No.226-2022-00028). The authors would like to thank Zepeng Li for his help with this work, including analysis and discussions.

References

Chen, J.; Yang, L.; Li, L.; Goodison, S.; and Sun, Y. 2022. Alignment-free comparison of metagenomics sequences via approximate string matching. *Bioinformatics Advances*, 2(1): vbac077.

Chothia, C.; and Lesk, A. M. 1986. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4): 823–826.

Corso, G.; Ying, Z.; Pándy, M.; Velickovic, P.; Leskovec, J.; and Liò, P. 2021. Neural Distance Embeddings for Biological Sequences. In *NeurIPS*, 18539–18551.

Dai, X.; Yan, X.; Zhou, K.; Wang, Y.; Yang, H.; and Cheng, J. 2020. Convolutional Embedding for Edit Distance. In *ACM SIGIR*, 599–608. ACM.

Forgy, E. W. 1965. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21: 768–769.

Fuglede, B.; and Topsøe, F. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *ISIT*, 31. IEEE.

Gao, L.; and Qi, J. 2007. Whole genome molecular phylogeny of large dsDNA viruses using composition vector method. *BMC evolutionary biology*, 7(1): 1–7.

Haubold, B.; Pfaffelhuber, P.; Domazet-Los$^{\smile}$ o, M.; and Wiehe, T. 2009. Estimating mutation distances from unaligned genomes. *Journal of Computational Biology*, 16(10): 1487–1500.

Hermans, A.; Beyer, L.; and Leibe, B. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

Kariin, S.; and Burge, C. 1995. Dinucleotide relative abundance extremes: a genomic signature. *Trends in genetics*, 11(7): 283–290.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86.

Leimeister, C.-A.; and Morgenstern, B. 2014. Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 30(14): 2000–2008.

Li, W.; and Godzik, A. 2021. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22, 1658–1659 (2006). *Scientific Reports*, 11: 3702.

Murtagh, F.; and Contreras, P. 2012. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining Knowl. Discov.*, 2(1): 86–97.

Needleman, S. B.; and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3): 443–453.

Ostrovsky, R.; and Rabani, Y. 2007. Low distortion embeddings for edit distance. *J. ACM*, 54(5): 23.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *IJCV*, 40(2): 99–121.

Sander, C.; and Schneider, R. 1991. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9(1): 56–68.

Sims, G. E.; Jun, S.-R.; Wu, G. A.; and Kim, S.-H. 2009. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106(8): 2677–2682.

Steinegger, M.; and Söding, J. 2018. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1): 1–8.

Ulitsky, I.; Burstein, D.; Tuller, T.; and Chor, B. 2006. The average common substring approach to phylogenomic reconstruction. *Journal of Computational Biology*, 13(2): 336–350.

von Luxburg, U. 2007. A tutorial on spectral clustering. *Stat. Comput.*, 17(4): 395–416.

Weinberger, K. Q.; and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2).

Zhang, X.; Yuan, Y.; and Indyk, P. 2019. Neural embeddings for nearest neighbor search under edit distance.

Zheng, W.; Yang, L.; Genco, R. J.; Wactawski-Wende, J.; Buck, M.; and Sun, Y. 2019. SENSE: Siamese neural network for sequence embedding and alignment-free comparison. *Bioinform.*, 35(11): 1820–1828.