Learning Genomic Sequence Representations using Graph Neural Networks over De Bruijn Graphs

Kacper Kapuśniak[*]
ETH Zürich, Switzerland
kkapusniak@student.ethz.ch

Manuel Burger
ETH Zürich, Switzerland
manuel.burger@inf.ethz.ch

Gunnar Rätsch
ETH Zürich, Switzerland
raetsch@inf.ethz.ch

Amir Joudaki
ETH Zürich, Switzerland
amir.joudaki@inf.ethz.ch

Abstract

The rapid expansion of genomic sequence data calls for new methods to achieve robust sequence representations. Existing techniques often neglect intricate structural details, emphasizing mainly contextual information. To address this, we developed k-mer embeddings that merge contextual and structural string information by enhancing De Bruijn graphs with structural similarity connections. Subsequently, we crafted a self-supervised method based on Contrastive Learning that employs a heterogeneous Graph Convolutional Network encoder and constructs positive pairs based on node similarities. Our embeddings consistently outperform prior techniques for Edit Distance Approximation and Closest String Retrieval tasks. [1]

1 Introduction

Genomic sequence data is expanding at an unparalleled pace, necessitating the development of innovative methods capable of providing accurate and scalable sequence representations [1]. These representations are foundational for many computational biology tasks, ranging from gene prediction to multiple sequence alignment [2]. The computational biology community has adopted methods from Natural Language Processing (NLP), such as Word2Vec and Transformers, to improve the representation of genomic sequences [3, 4, 5, 6, 7]. These NLP-based approaches are adept at capturing context within the sequence, a vital aspect where the semantics of words often outweigh the precise letters composing them.

To capture structural nuances, one might consider character-level n-gram models. However, a uniform representation of each n-gram across all sequences can oversimplify the problem, and on the other hand, applying techniques like transformer-based models on n-grams can escalate computational demands. Consequently, these methods may overlook nuanced k-mer variations essential for comprehending single-nucleotide polymorphisms (SNPs) and other minor sequence changes. These SNPs influence disease susceptibility, phenotypic traits, and drug responses [8, 9, 10].

Therefore, we introduced a k-mer embedding approach that combines metagenomic context and string structure. In our methodology, contextual information refers to the relationships between k-mers situated closely within sequences, while structural information examines nucleotide patterns within a k-mer and their relations to other k-mers. Using this, we constructed a metagenomic graph that builds upon the De Bruijn Graph to capture not only the transitions of k-mers but also the structural similarities.

[*]The author is now affiliated with the University of Oxford: kacper.kapusniak@keble.ox.ac.uk.
[1]Code Available at https://github.com/ratschlab/genomic-gnn.git

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

Given the advances in Graph Neural Networks (GNNs), e.g. by Kipf and Welling [11], we grounded our method in GNNs but designed for heterogeneous graphs. This approach effectively recognizes and uses both contextual and structural connection types. Further, drawing from the success of self-supervised pre-training in Natural Language Processing and Computer Vision [12, 13, 14], we designed a self-supervised objective for genomic graph data. We employed contrastive loss aiming to align closely in representation space k-mers with similar context and structure.

Finally, we benchmarked our technique on two downstream tasks: Edit Distance Approximation and Closest String Retrieval, influenced by Corso *et al.* [15]. The former estimates the minimum number of changes required to transform one genomic sequence into the other but without quadratic computational complexity. This is crucial, as understanding the evolutionary distance between constantly evolving sequences remains a primary challenge in biology. The latter task, Closest String Retrieval, involves efficiently finding sequences resembling a provided query, a method biologists use when categorizing new genes.

2 Related Work

Genomic Sequence Representation Machine learning methods have emerged in computational biology to represent genomic sequences. A key component is the k-mer: a continuous nucleotide sequence of length k. The Word2Vec method [16], which represents words as vectors using their context, treats overlapping k-mers in genomic sequences as words in sentences. Building on this, Ren *et al.* [5] introduced *kmer2vec* to apply Word2Vec to genomic data for Multiple Sequence Alignment. Another strategy is to use the De Bruijn graph, where k-mers are nodes and their overlaps are edges, in conjunction with Node2Vec [17], which derives node features from the contextual information of biased random walks. This method underpins Narayanan *et al.* [18]'s *GRaDL* for early animal genome disease detection. K-mers also pair well with transformer-based models: Ji *et al.* [4]'s *DNABERT* leverages a BERT-inspired objective [12] and k-mer tokenization to predict genome-wide regulatory elements. Similarly, Aakur *et al.* [6]'s *Metagenome2Vec* blends Node2Vec with transformers to analyze metagenomes with limited labeled data. Given the high computational demands of these transformer-based approaches, they fall outside the scope of our benchmarks in this study.

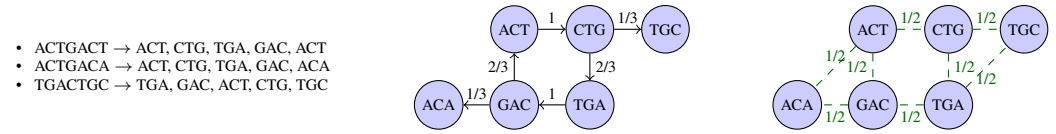
Graph Neural Networks Graph Convolutional Networks (GCNs) are foundational to several innovative applications in graph-based machine learning [11]. In genomics, GNNs have been applied in metagenomic binning; for instance, Lamurias *et al.* [19]. As we aim to enhance our node embeddings with structural similarity, both heterogeneity and heterophily are key considerations. Recognizing the ubiquity of heterogeneity in real-world graphs, Relational GCNs (R-GCNs) were developed. These networks expand upon GCNs by generalizing the convolution operation to handle different edge types, assigning distinct learnable weight matrices for each relation type [20]. To tackle heterophily, where distant nodes in a graph may bear similar features, Pei *et al.* [21] introduced Geom-GCN, which maps nodes to a latent space, while Zhu *et al.* [22] suggested a distinct encoding approach for node embeddings and neighborhood aggregations.

Self-Supervised Learning Self-supervised learning (SSL) enables effective use of unlabeled data and reduces dependence on annotated labels [23]. Among SSL methods, contrastive learning has made a significant impact [13, 14]. At its core, contrastive learning seeks to bring similar data instances closer in the embedding space while pushing dissimilar ones apart. When applied to graph data, several techniques have been proposed for obtaining positive pairs, including uniform sampling, node dropping, and random walk sampling [24, 25, 26].

3 Methodology

3.1 Metagenomic Graph

The De Bruijn Graph, constructed from metagenomic sequences, forms the foundation of our method. In this graph, each k-mer, a substring of length k derived from the sequences, is uniquely represented by a different node. Additionally, an edge from node v_i to node v_j in the graph indicates that the k-mer at node v_i directly precedes the k-mer at node v_j in one of the sequences of the metagenome.



(a) Sequences with Derived K-mers (b) De Bruijn Graph's edges (c) Sub-k-mer Frequency edges

Figure 1: Example of a Metagenomic Graph where nodes represent k-mers, De Bruijn Graph's edges capture contextual information, while Sub-k-mer Frequency edges depict structural similarity.

When used, edge weights represent the frequency of these transitions, thereby capturing intrinsic genomic structures within the graph.

Although Node2Vec effectively captures the sequential context in De Bruijn graphs, it overlooks structural k-mer similarities. To address this, we expand the graph to include connections based on these similarities, complementing the transition probabilities. In subsequent sections, we detail the formulation of the two edge types for our graph, where nodes $\{v_i, v_j, \dots\}$ represent k-mers, as depicted in Figure 1.

De Bruijn Graph's edges The first edge type is designed to capture contextual information. Let $T(v_i, v_j)$ represent the count of transitions between k-mers within a dataset of genomic sequences. The weight of an edge connecting nodes v_i and v_j, $w_{ij}^{(dBG)}$, is defined by,

$$w_{ij}^{(dBG)} = \frac{T(v_i, v_j)}{\sum_{l \in \delta^+(v_i)} T(v_i, v_l)}, \tag{1}$$

where $\delta^+(v_i)$ denotes nodes adjacent to v_i via outgoing edges.

Sub-k-mer Frequency edges To efficiently capture the structural similarity between strings, we introduce a method using sub-k-mer frequency vectors, denoted as $\mathbf{y}^{(\text{KF}_{\text{sub\_k}})}$. This vector quantifies the occurrences of each sub-k-mer of length sub_k within a given k-mer. Specifically, the i-th entry indicates the frequency of the i-th sub-k-mer,

$$\mathbf{y}^{(\text{KF}_{\text{sub\_k}})}[i] = \sum_{j=1}^{k-\text{sub\_k}+1} \mathbf{1}_{\text{k-mer}[j:j+\text{sub\_k}-1]=s_i}, \forall i, s_i \in \Sigma^{\text{sub\_k}}. \tag{2}$$

The k-mer similarity is then determined using the cosine similarity between these sub-k-mer frequency vectors,

$$w_{ij}^{(\text{KF}_{\text{sub\_k}})} = \frac{\mathbf{y}_i^{(\text{KF}_{\text{sub\_k}})^T} \mathbf{y}_j^{(\text{KF}_{\text{sub\_k}})}}{\|\mathbf{y}_i^{(\text{KF}_{\text{sub\_k}})}\|_2 \|\mathbf{y}_j^{(\text{KF}_{\text{sub\_k}})}\|_2}. \tag{3}$$

This method, scaling linearly with the frequency vector size per weight, provides a computational advantage in practice over the direct Edit Distance calculation for k-mers, which exhibits k^2 complexity per weight. Even so, computation for each node pair remains necessary. Given that node counts might approach 4^k, we apply edge-filtering at threshold t, retaining only the links with the highest similarity. The filtered set of weights is then,

$$\mathbf{W}^{(\text{KF}_{\text{sub\_k}})} = \{\, w_{ij}^{(\text{KF}_{\text{sub\_k}})} \mid w_{ij}^{(\text{KF}_{\text{sub\_k}})} \geq t \,\}. \tag{4}$$

To better accommodate graphs for larger k values, we have also developed a more scalable approximation of the above approach. It utilizes state-of-the-art approximate nearest neighbor search [27] on the sub-k-mer frequency vectors, replacing the computationally demanding pairwise cosine similarity calculations. The details of this adaptation and its experimental results are outlined in Appendix A, demonstrating the method's effectiveness in processing large metagenomic graphs.

Notation The metagenomic graph is formally defined as $G = (V, E, W)$. In this graph, nodes V correspond to individual k-mers. The edges E can be categorized into two sets: De Bruijn Graphs's edges $E^{(dBG)}$ and Sub-k-mer Frequency edges $E^{(KF)}$. Edges in $E^{(KF)}$ may be further subdivided based on various sub_k values. Thus, edge weights W can contain $\mathbf{W}^{(dBG)}$ and several $\mathbf{W}^{(KF_{\text{sub\_k}})}$.

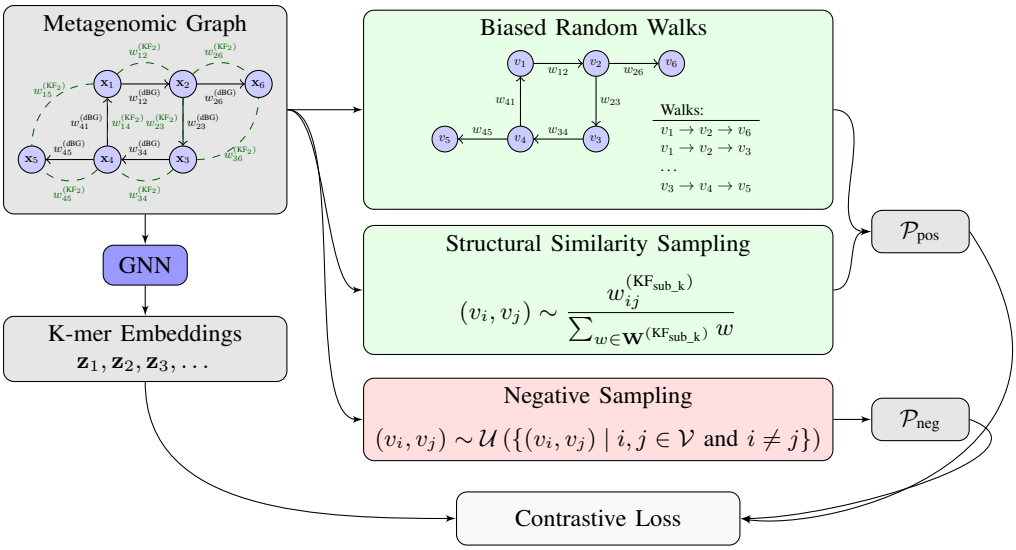


Figure 2: Self-Supervised Graph Contrastive Learning approach.

3.2 Encoder

We tailored GNNs for a heterogeneous metagenomic graph to capture nuanced k-mer relationships. The design employs varying depths of message passing: deeper for De Bruijn edges to capture broader context and shallower for similarity measures. Central to this GNN is the adapted Graph Convolutional Layer, formulated as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{(\text{edge\_type})^{-\frac{1}{2}}} \tilde{\mathbf{W}}^{(\text{edge\_type})} \tilde{\mathbf{D}}^{(\text{edge\_type})^{-\frac{1}{2}}} \mathbf{H}^{(l)} \mathbf{\Theta}^{(l)}\right), \tag{5}$$

where $\tilde{\mathbf{W}}^{(\text{edge\_type})}$ includes added self-loops and $\tilde{\mathbf{D}}_{ii}$ is its diagonal degree matrix. The term *edge_type* refers to either dBG or $KF_{\text{sub\_k}}$. The GCN layout consists of multiple layers, each characterized by a unique edge feature type and the number of channels.

3.3 Self-Supervised Task

We investigate the use of a contrastive learning method for k-mer representations. Graph nodes are initialized using a sub-k-mer frequency vector. Positive and negative pairs are sampled and, along with the k-mer representations from the encoder, are used to compute the loss, as depicted in Figure 2.

Biased Random Walk Sampling We employ Biased Random Walk Sampling to capture k-mer contextual information. This approach uses $w^{(dBG)}$ edges to conduct walks, implemented exactly as in Node2Vec [17]. Given a walk of a set length, we extract positive pairs by applying a window of size m. Using a shrink factor δ, drawn uniformly from $\{1, \dots, m\}$, we determine the range $i \pm \delta$ within which nodes are considered positive pairs to node v_i. Repeating this across multiple random walks, we gather a comprehensive set of positive pairs.

Structural Similarity Sampling To capture the structural notion of k-mers, we sample pairs with probability proportional to sub-k-mer frequency similarity, $w^{(\text{KF}_{\text{sub\_k}})}$. The aim is for k-mers linked by higher similarity (weights closer to 1) to have similar representations. The probability of sampling is given by,

$$(v_i, v_j) \sim \frac{w_{ij}^{(\text{KF}_{\text{sub\_k}})}}{\sum_{w \in \mathbf{W}^{(\text{KF}_{\text{sub\_k}})}} w}. \tag{6}$$

Negative Sampling We randomly select negative pairs from all node pairs in the graph, leveraging the assumption that most such pairs lack a high similarity edge. This approach ensures diversity in learned representations.

Loss Function Having established both positive $\mathcal{P}_{\text{pos}}$ and negative $\mathcal{P}_{\text{neg}}$ pair types, we apply the contrastive loss function. Following the approach by Hamilton *et al.* [28], and using $\sigma(x)$ as the sigmoid function, the equation is,

$$l_{ij} = -\log\left(\sigma\left(\mathbf{z}_i^T \mathbf{z}_j\right)\right) - \sum_{(i,l)\in\mathcal{P}_{\text{neg}}(i)} \log\left(\sigma\left(-\mathbf{z}_i^T \mathbf{z}_l\right)\right). \tag{7}$$

To reduce memory usage, we employed Neighborhood Sampling, again from [28], for mini-batching during training.

4 Bioinformatics Tasks

Edit Distance Approximation The task aims to calculate the edit distance without the burden of quadratic complexity. The *NeuroSEED* framework by Corso *et al.* [15] offers a solution by providing sequence representations trained on a ground truth set of edit distances. In our experimental approach, we began with sequence representations derived from k-mer embeddings and subsequently fine-tuned them with a single linear layer. Our experiments were tested against One-Hot encoding (for $k = 1$ corresponding to *NeuroSEED* [15]), Word2Vec, and Node2Vec. To find optimal hyperparameters, we executed a grid search on the validation set. Based on Corso *et al.* [15] 's findings, we employed the hyperbolic function as it consistently outperformed other distance measures. Our primary metric for evaluation was the percentage Root Mean Squared Error (% RMSE), where l denotes the dataset's maximum sequence length, h represents the hyperbolic distance function, and f_θ indicates the downstream model,

$$\%\text{RMSE}(\theta, \mathcal{D}) = \frac{100}{l}\sqrt{\sum_{s_1,s_2\in\mathcal{D}} \left(\text{EditDistance}(s_1, s_2) - l \cdot h\left(f_\theta(\boldsymbol{z_1}), f_\theta(\boldsymbol{z_2})\right)\right)^2}. \tag{8}$$

Closest String Retrieval The task is to find the sequence from a reference set that is closest to a specified query. We assessed embeddings fine-tuned on the edit distance approximation task using Convolutional Neural Networks (CNNs). These embeddings were contrasted with ones directly derived from our Self-supervised method, One-Hot, Word2Vec, or Node2Vec, through concatenation or taking the mean of k-mer embeddings. For performance assessment, we used top-$n\%$ accuracies, measuring how often the actual sequence appears within the top $n\%$ of positions based on the closeness of embedding vectors in hyperbolic space. We selected the optimal model for the embeddings based on the validation loss observed for the previous Edit Distance task.

5 Results and Analysis

In all our experiments, the memory requirements of the One-Hot method increase exponentially, leading to its exclusion from our results for $k > 7$. When pre-training exclusively on the training set, our method, thanks to the GCN encoder, can generalize beyond k-mers present in the training

Table 1: RMSE ↓ for the Edit Distance Approximation Task fine-tuned with Single Linear Layer. The best result per k is highlighted in bold, with deeper green shades indicating better performance across all runs. The standard deviation is based on three runs.

	RT988 dataset				Qiita dataset			
k	One-Hot	Word2Vec	Node2Vec	Our CL	One-Hot	Word2Vec	Node2Vec	Our CL
1	$\mathbf{0.43 \pm 0.01}$	0.44 ± 0.01	0.50 ± 0.01	-	$\mathbf{2.46 \pm 0.03}$	2.57 ± 0.01	2.66 ± 0.04	-
2	$\mathbf{0.40 \pm 0.01}$	0.42 ± 0.01	0.45 ± 0.01	$\mathbf{0.40 \pm 0.01}$	2.31 ± 0.01	2.22 ± 0.03	2.43 ± 0.01	$\mathbf{2.14 \pm 0.02}$
3	0.41 ± 0.01	0.42 ± 0.01	0.38 ± 0.01	$\mathbf{0.37 \pm 0.01}$	2.41 ± 0.01	2.29 ± 0.01	2.29 ± 0.04	$\mathbf{2.09 \pm 0.03}$
4	0.42 ± 0.01	0.41 ± 0.01	0.38 ± 0.01	$\mathbf{0.37 \pm 0.01}$	2.65 ± 0.03	2.35 ± 0.01	2.27 ± 0.04	$\mathbf{2.00 \pm 0.01}$
5	0.43 ± 0.01	0.40 ± 0.01	0.36 ± 0.01	$\mathbf{0.35 \pm 0.01}$	3.14 ± 0.01	2.17 ± 0.01	2.16 ± 0.02	$\mathbf{2.00 \pm 0.01}$
6	0.43 ± 0.01	0.39 ± 0.01	0.37 ± 0.01	$\mathbf{0.36 \pm 0.01}$	3.51 ± 0.04	2.00 ± 0.01	2.12 ± 0.03	$\mathbf{1.97 \pm 0.01}$
7	0.44 ± 0.01	-	-	$\mathbf{0.36 \pm 0.01}$	4.12 ± 0.07	-	-	$\mathbf{1.99 \pm 0.01}$
8	-	-	-	$\mathbf{0.35 \pm 0.01}$	-	-	-	$\mathbf{1.96 \pm 0.01}$

set. In contrast, Node2Vec and Word2Vec can only handle k-mer sizes up to the diversity of the training dataset. Hence, for $k > 6$, where the test set introduces new k-mers, we had to exclude these methods.

5.1 Edit Distance Approximation

Table 1 presents the results obtained using our pre-trained embeddings to estimate edit distances between sequences on the *RT988* and *Qiita* datasets from [15]. For the *RT988* dataset, our Contrastive Learning (CL) and Node2Vec techniques surpass Word2Vec and One-Hot. The increased losses in *Qiita* highlight its greater complexity. In this context, our method's integration of k-mer structural similarity becomes even more beneficial, outperforming all other tested methods. This benefit becomes more evident as k increases, underscoring our embedding's capability to adapt to new nodes.

5.2 Closest String Retrieval

Tables 2a and 2b present the performance of our zero-shot sequence embeddings, directly derived from the aggregation of our k-mer embeddings, in retrieving the nearest sequences in the *Qiita* dataset from [15]. The tables also showcase a comparison with the embeddings that were specifically fine-tuned for the Edit Distance Task, a process outlined by Corso *et al.* [15].

For direct k-mer aggregation, our Contrastive Learning (CL) embeddings are obtained through concatenation, while for k-mer aggregation with One-Hot, Word2Vec, and Node2Vec, we report the results of the better performing method, either concatenation or averaging. The superior zero-shot non-parametric retrieval performance of our CL method emphasizes the combined utility of both

Table 2: Mean Top retrieval performance. Best results per k are highlighted in bold, with deeper green shades indicating better performance separately for Zero-Shot and Fine-Tuned. The standard deviation is based on three runs.

(a) Top 1% ↑

	Zero-Shot: Aggregated K-mer Embeddings				Fine-Tuned: *NeuroSEED* with K-mer Embeddings (equivalent to Corso *et al.* [15] for One-Hot at $k = 1$)			
k	One-Hot	Word2Vec	Node2Vec	Our CL	One-Hot	Word2Vec	Node2Vec	Our CL
1	**50.3**	45 ± 0.1	45.3 ± 1.5	-	46.9 ± 0.9	47.5 ± 1.3	46.2 ± 0.1	-
2	49.9	46.7 ± 0.1	49.9 ± 0.6	**52.2 ± 0.7**	48 ± 0.1	47.8 ± 1.8	47.6 ± 0.5	48.3 ± 0.7
3	46.8	48.6 ± 0.1	51.2 ± 0.2	**53.1 ± 0.4**	46.4 ± 1	47.3 ± 0.1	49.1 ± 0.6	48.2 ± 0.3
4	45.3	46.9 ± 0.1	49.8 ± 0.2	**53.3 ± 0.3**	46.2 ± 0.5	46.8 ± 1.5	48.2 ± 1.3	47.7 ± 0.8
5	45.4	42.3 ± 0.1	50 ± 0.4	**50.5 ± 0.1**	46.6 ± 0.8	47 ± 1.1	48.8 ± 1.8	47.8 ± 0.3
6	46.3	41.3 ± 0.1	49.6 ± 0.3	50 ± 0.7	48.3 ± 1.2	45.2 ± 0.5	**50.1 ± 0.4**	47 ± 0.9
7	44.5	-	-	48.3 ± 1.1	44.8 ± 0.5	-	-	**48.9 ± 0.6**
8	-	-	-	**50.2 ± 0.1**	-	-	-	48 ± 0.3

(b) Top 10% ↑

	Zero-Shot: Aggregated K-mer Embeddings				Fine-Tuned: *NeuroSEED* with K-mer Embeddings (equivalent to Corso *et al.* [15] for One-Hot at $k = 1$)			
k	One-Hot	Word2Vec	Node2Vec	Our CL	One-Hot	Word2Vec	Node2Vec	Our CL
1	60.1	58.0 ± 0.1	60.5 ± 0.3	-	**75.9 ± 1.2**	75.2 ± 0.9	74.9 ± 0.7	-
2	60.7	60.3 ± 0.1	60.1 ± 0.4	68.0 ± 1.1	75.4 ± 0.1	76 ± 0.8	75.3 ± 0.7	**76.4 ± 0.5**
3	61.4	61.3 ± 0.1	64.9 ± 0.2	70.8 ± 0.6	75.2 ± 0.4	75 ± 0.6	75.4 ± 0.3	**75.6 ± 0.6**
4	64.6	64.4 ± 0.3	73.8 ± 0.3	**78.1 ± 0.1**	74.6 ± 0.1	75.2 ± 0.5	75.1 ± 1.6	75.4 ± 0.3
5	68.5	62.7 ± 0.2	74.3 ± 0.2	**79.5 ± 0.2**	75.8 ± 0.3	75.3 ± 0.8	76 ± 1.9	75.3 ± 0.8
6	68.4	67.3 ± 0.1	71.4 ± 0.3	**81.3 ± 0.1**	75.6 ± 0.1	74.1 ± 0.9	78.1 ± 1.0	74.8 ± 0.7
7	65.6	-	-	**80.1 ± 0.5**	71.2 ± 0.3	-	-	77.8 ± 0.5
8	-	-	-	**79.4 ± 0.2**	-	-	-	75.4 ± 0.4

context and structural similarity during self-supervised pre-training. Notably, while k-mers of size around three are optimal for Top 1% retrieval, larger k-mers excel in the Top 10% metrics. This suggests that smaller k-mers are better at discerning local sequence distances, whereas larger ones capture broader sequence distances.

For embeddings fine-tuned using CNNs for Edit Distance Approximation, the complexity of CNNs appears to obscure differences between the embeddings. Notably, our method based solely on zero-shot concatenated k-mer embeddings outperforms this complex fine-tuning. This shows the clear advantage of our embeddings over the *NeuroSEED* method by Corso *et al.* [15].

6 Conclusion

In our study, we introduced a novel k-mer embedding technique that seamlessly integrates metagenomic contextual and structural nuances, achieved through the enhancement of the De Bruijn graph and the use of contrastive learning. In the Edit Distance Approximation task, our technique consistently demonstrated superior performance compared to One-Hot, Word2Vec, and Node2Vec. Moreover, without requiring any downstream fine-tuning, our aggregated k-mer embeddings outperformed the *Neuroseed* method by Corso *et al.* [15] in the Closest String Retrieval task. These findings suggest potential broader uses in computational biology.

7 Acknowledgements

Amir Joudaki is funded through Swiss National Science Foundation Project Grant #200550 to Andre Kahles, and partially funded by ETH Core funding award to Gunnar Ratsch. Manuel Burger is funded by grant #2022-278 of the Strategic Focus Area "Personalized Health and Related Technologies (PHRT)" of the ETH Domain (Swiss Federal Institutes of Technology).

References

[1] Z. D. Stephens *et al.*, "Big data: Astronomical or genomical?" *PLoS biology*, vol. 13, no. 7, e1002195, 2015.

[2] W. S. Alharbi and M. Rashid, "A review of deep learning applications in human genomics using next-generation sequencing data," *Human Genomics*, vol. 16, no. 1, pp. 1–20, 2022.

[3] P. Ng, "Dna2vec: Consistent vector representations of variable-length k-mers," *arXiv preprint arXiv:1701.06279*, 2017.

[4] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, "Dnabert: Pre-trained bidirectional encoder representations from transformers model for dna-language in genome," *Bioinformatics*, vol. 37, no. 15, pp. 2112–2120, 2021.

[5] R. Ren, C. Yin, and S. S.-T. Yau, "Kmer2vec: A novel method for comparing dna sequences by word2vec embedding," *Journal of Computational Biology*, vol. 29, no. 9, pp. 1001–1021, 2022.

[6] S. N. Aakur *et al.*, "Metagenome2vec: Building contextualized representations for scalable metagenome analysis," in *2021 International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2021, pp. 500–507.

[7] G. Fang, F. Zeng, X. Li, and L. Yao, "Word2vec based deep learning network for dna n4-methylcytosine sites identification," *Procedia Computer Science*, vol. 187, pp. 270–277, 2021.

[8] A. F. Wright, "Genetic variation: Polymorphisms and mutations," *e LS*, 2001.

[9] F. S. Collins, L. D. Brooks, and A. Chakravarti, "A dna polymorphism discovery resource for research on human genetic variation," *Genome research*, vol. 8, no. 12, pp. 1229–1231, 1998.

[10] C. Rodríguez-Antona and M. Taron, "Pharmacogenomic biomarkers for personalized cancer treatment," *Journal of internal medicine*, vol. 277, no. 2, pp. 201–217, 2015.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[12] 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.

[13] J.-B. Grill *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

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

[15] G. Corso, Z. Ying, M. Pándy, P. Veličković, J. Leskovec, and P. Liò, "Neural distance embeddings for biological sequences," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 539–18 551, 2021.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[17] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[18] S. Narayanan, A. Ramachandran, S. N. Aakur, and A. Bagavathi, "Gradl: A framework for animal genome sequence classification with graph representations and deep learning," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2020, pp. 1297–1303.

[19] A. Lamurias, M. Sereika, M. Albertsen, K. Hose, and T. D. Nielsen, "Metagenomic binning with assembly graph embeddings," *Bioinformatics*, vol. 38, no. 19, pp. 4481–4487, 2022.

[20] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, Springer, 2018, pp. 593–607.

[21] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," *arXiv preprint arXiv:2002.05287*, 2020.

[22] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in neural information processing systems*, vol. 33, pp. 7793–7804, 2020.

[23] X. Liu *et al.*, "Self-supervised learning: Generative or contrastive," *IEEE transactions on knowledge and data engineering*, vol. 35, no. 1, pp. 857–876, 2021.

[24] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International conference on machine learning*, PMLR, 2020, pp. 4116–4126.

[25] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.

[26] J. Qiu *et al.*, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1150–1160.

[27] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

A Scalable K-mer Graph Construction

This appendix addresses the challenge of assembling k-mer graphs for larger k values, offering a method more efficient than the pairwise cosine similarity calculations in our original framework. We incorporate the FAISS library [27], which uses an inverted file structure for efficient approximate nearest neighbor searches. This library identifies a predetermined number of nearest neighbors for each node in the metagenomic graph, forming Sub-k-mer Frequency edges weighted according to distance metrics from the search. The process can be used for more than one sub-k value, potentially generating several subtypes of edges.

The effectiveness of this extension is demonstrated in edit distance approximation and closest string retrieval tasks, as presented in Tables 3 and 4. Performance for larger k-mers ($k = 10$ and $k = 15$) is consistent with that of smaller k-mers, matching or surpassing Node2Vec and Word2Vec benchmarks. Unlike these benchmarks, our method effectively generates embeddings for new k-mers, unseen in the training data, thus facilitating scalability for larger k. However, performance declines for very large k-mers ($k = 20$ and $k = 30$). This decline is likely due to the high uniqueness of k-mers at these sizes in our datasets, reducing the informativeness of transition probabilities and the relevance of graph-based structural similarities.

Table 3: RMSE ↓ for the Edit Distance Approximation Task on larger k, fine-tuned with a single linear layer. Results derived using our contrastive learning framework with approximate nearest neighbor search instead of cosine similarity. The standard deviation is based on three runs.

	RT988 dataset	Qiita dataset
k	Our CL	Our CL
10	0.36 ± 0.01	2.05 ± 0.01
15	0.36 ± 0.01	2.01 ± 0.02
20	0.36 ± 0.01	2.12 ± 0.01
30	0.37 ± 0.01	2.36 ± 0.02

Table 4: Mean Top retrieval performance on larger k. Results derived using our contrastive learning framework with approximate nearest neighbor search instead of cosine similarity. The standard deviation is based on three runs.

(a) Top 1% ↑

	Zero-Shot: Aggregated K-mer Embeddings	Fine-Tuned: *NeuroSEED* with K-mer Embeddings
k	Our CL	Our CL
10	49.3 ± 0.4	46.5 ± 0.4
15	48.7 ± 0.6	46.0 ± 0.6
20	44.0 ± 0.8	43.8 ± 0.5
30	41.1 ± 0.9	42.1 ± 0.6

(b) Top 10% ↑

	Zero-Shot: Aggregated K-mer Embeddings	Fine-Tuned: *NeuroSEED* with K-mer Embeddings
k	Our CL	Our CL
10	77.6 ± 1.2	71.9 ± 0.5
15	78.4 ± 0.8	71.9 ± 0.4
20	76.1 ± 0.8	70.8 ± 0.4
30	71.3 ± 0.9	69.1 ± 0.9

B Analysis of Sampling Techniques in Contrastive Learning

Table 5 presents the impact of different graph edges and corresponding sampling methods on the Edit Distance Approximation task. The results are presented for three distinct scenarios: training exclusively with dBG edges using Biased Random Walk Sampling, training exclusively with KF edges using Structural Similarity Sampling, and the standard approach that combines both edge types. These results highlight that integrating contextual and structural knowledge yields superior performance compared to employing each sampling strategy separately.

Table 5: RMSE ↓ for the Edit Distance Approximation Task. The standard deviation is based on three runs.

	Qiita dataset		
k	dBG edges only	KF edges only	Both Edge Types
3	2.14 ± 0.01	2.12 ± 0.02	$\mathbf{2.09 \pm 0.03}$
6	2.19 ± 0.03	2.13 ± 0.01	$\mathbf{1.97 \pm 0.01}$
8	2.04 ± 0.01	2.09 ± 0.01	$\mathbf{1.96 \pm 0.01}$

C Analysis of Graph Autoencoder as an Alternative Self-Supervised Task to Contrastive Learning

This appendix outlines an alternative self-supervised learning task that applies a Graph Autoencoder (GAE) to the same Metagenomic Graph as the original method. This task eliminates the need for sampling, potentially offering computational benefits, especially for large graphs.

Figure 3 illustrates the GAE methodology. After the encoding stage, our model employs two types of decoders: an edge decoder and a node decoder. Both are designed with simplicity in mind to avoid overfitting.

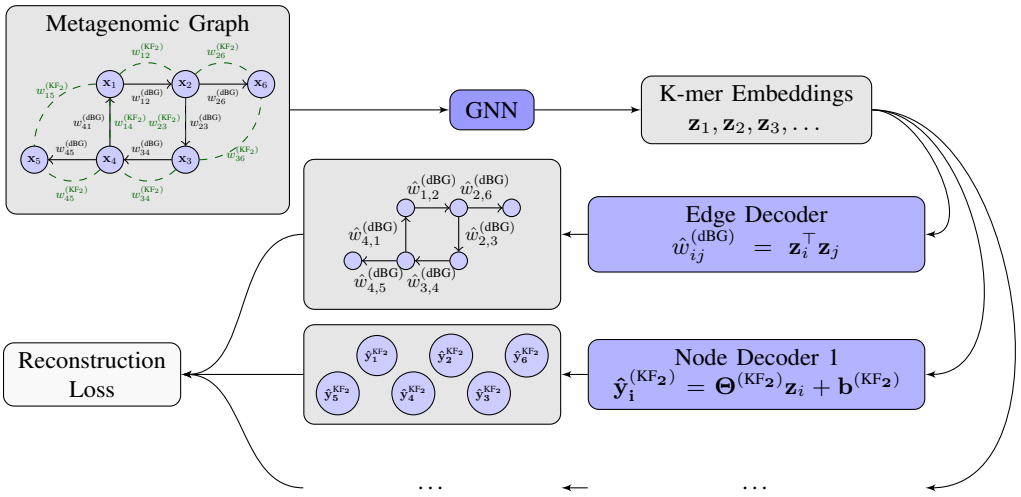


Figure 3: Graph Autoencoder approach.

Edge Decoder The edge decoder's role is to capture contextual relationships from the original De Bruijn graph by reconstructing its transition probabilities. We employ an inner product decoder, defined as,

$$\hat{w}_{ij}^{(dBG)} = \mathbf{z}_i^T \mathbf{z}_j.$$

Node Decoders The node decoder's role is to create embeddings reflecting structural similarities among k-mers by reconstructing the sub-k-mer frequency vectors for each node. Our model works

Table 6: RMSE ↓ for the Edit Distance Approximation Task fine-tuned with Single Linear Layer. The standard deviation is based on three runs.

	RT988 dataset		Qiita dataset	
k	Our GAE	Our CL	Our GAE	Our CL
2	0.40 ± 0.01	0.40 ± 0.01	2.34 ± 0.01	$\mathbf{2.14 \pm 0.02}$
3	0.37 ± 0.01	0.37 ± 0.01	2.10 ± 0.03	$\mathbf{2.09 \pm 0.03}$
4	0.37 ± 0.01	0.37 ± 0.01	2.09 ± 0.01	$\mathbf{2.00 \pm 0.01}$
5	0.36 ± 0.01	$\mathbf{0.35 \pm 0.01}$	2.09 ± 0.03	$\mathbf{2.00 \pm 0.01}$
6	$\mathbf{0.35 \pm 0.01}$	0.36 ± 0.01	2.05 ± 0.05	$\mathbf{1.97 \pm 0.01}$
7	0.36 ± 0.01	0.36 ± 0.01	$\mathbf{1.96 \pm 0.02}$	1.99 ± 0.01
8	0.36 ± 0.01	$\mathbf{0.35 \pm 0.01}$	2.06 ± 0.01	$\mathbf{1.96 \pm 0.01}$

with more than one sub_k edge subtype, training a separate decoder for each. Each decoder consists of a linear layer, formulated as,

$$\hat{\mathbf{y}}_{\mathbf{i}}^{(\text{KF}_{\text{sub\_k}})} = \mathbf{\Theta}^{(\text{KF}_{\text{sub\_k}})}\mathbf{z}_i + \mathbf{b}^{(\text{KF}_{\text{sub\_k}})},$$

with separate parameters $\mathbf{\Theta}^{(\text{KF}_{\text{sub\_k}})}$ and $\mathbf{b}^{(\text{KF}_{\text{sub\_k}})}$ for each sub_k.

Reconstruction Loss In our GAE model, the L1 loss is used for the edge decoder, while the Mean Squared Error is applied to the node decoders. The total loss, denoting the set of all sub_ks as K and the total number of nodes as N, is calculated as follows,

$$L_{\text{GAE}} = \sum_{(i,j)\in E} |w_{ij}^{(dBG)} - \hat{w}_{ij}^{(dBG)}| + \frac{1}{|K|} \sum_{\text{sub\_k}\in K} \sum_{i\in \text{N}} (\mathbf{y}_{\mathbf{i}}^{(\text{KF}_{\text{sub\_k}})} - \hat{\mathbf{y}}_{\mathbf{i}}^{(\text{KF}_{\text{sub\_k}})})^2.$$

Results The results of applying the GAE to the Edit Distance Approximation and Closest String Retrieval tasks are presented in Tables 6 and 7. Although the GAE demonstrates overall good performance, often surpassing Word2Vec and Node2Vec, Contrastive Learning consistently achieves superior results.

Table 7: Mean Top retrieval performance. The standard deviation is based on three runs.

(a) Top 1% ↑

	Zero-Shot: Aggregated K-mer Embeddings		Fine-Tuned: *NeuroSEED* with K-mer Embeddings	
k	Our GAE	Our CL	Our GAE	Our CL
2	47.4 ± 0.6	$\mathbf{52.2 \pm 0.7}$	47.9 ± 1.1	48.3 ± 0.7
3	50.1 ± 0.3	$\mathbf{53.1 \pm 0.4}$	47.9 ± 0.7	48.2 ± 0.3
4	49.4 ± 0.5	$\mathbf{53.3 \pm 0.3}$	49.3 ± 0.5	47.7 ± 0.8
5	50.0 ± 0.2	$\mathbf{50.5 \pm 0.1}$	49.4 ± 0.2	47.8 ± 0.3
6	$\mathbf{50.2 \pm 0.7}$	50 ± 0.7	48.5 ± 1.4	47 ± 0.9
7	$\mathbf{49.8 \pm 0.3}$	48.3 ± 1.1	49.4 ± 0.7	48.9 ± 0.6
8	45.0 ± 0.1	$\mathbf{50.2 \pm 0.1}$	49.4 ± 1.2	48 ± 0.3

(b) Top 10% ↑

	Zero-Shot: Aggregated K-mer Embeddings		Fine-Tuned: *NeuroSEED* with K-mer Embeddings	
k	Our GAE	Our CL	Our GAE	Our CL
2	62.6 ± 0.2	68.0 ± 1.1	75.4 ± 0.3	$\mathbf{76.4 \pm 0.5}$
3	68.1 ± 0.4	70.8 ± 0.6	75.4 ± 0.8	$\mathbf{75.6 \pm 0.6}$
4	75.1 ± 0.3	$\mathbf{78.1 \pm 0.1}$	75.3 ± 0.3	75.4 ± 0.3
5	78.5 ± 0.5	$\mathbf{79.5 \pm 0.2}$	75.4 ± 119	75.3 ± 0.8
6	80.4 ± 0.4	$\mathbf{81.3 \pm 0.1}$	75.2 ± 0.6	74.8 ± 0.7
7	$\mathbf{81.3 \pm 0.5}$	80.1 ± 0.5	75.6 ± 1.0	77.8 ± 0.5
8	70.9 ± 0.2	$\mathbf{79.4 \pm 0.2}$	76.5 ± 0.5	75.4 ± 0.4