Neural Distance Embeddings for Biological Sequences

NeurIPS 2021

Presenter: Joung Min Choi

Background

- **Edit distance** $D(s_1, s_2)$
  - The evolutionary distance between pairs of sequences
  - The minimum number of insertions, deletions or substitutions required to transform a string s_1 into another string s_2

- **Bioinformatics tasks**

  1) Edit distance approximation
  - The task of finding the distance or similarity between two strings

  2) Hierarchical clustering
  - Discovering the intrinsic hierarchical structure given by evolutionary history
  - Given a pairwise distance function, defining a tree with internal points corresponding to clusters and leaves to datapoints

Background

3) Multiple sequence alignment (MSA)

- Aligning three or more sequences is used for the identification of active and binding sites
- Finding the global optimum of N sequences: NP-complete
- Related task: Approximation of the Steiner string

  ⇒ Minimizing the sum of the distances (consensus error) to the sequences in a set

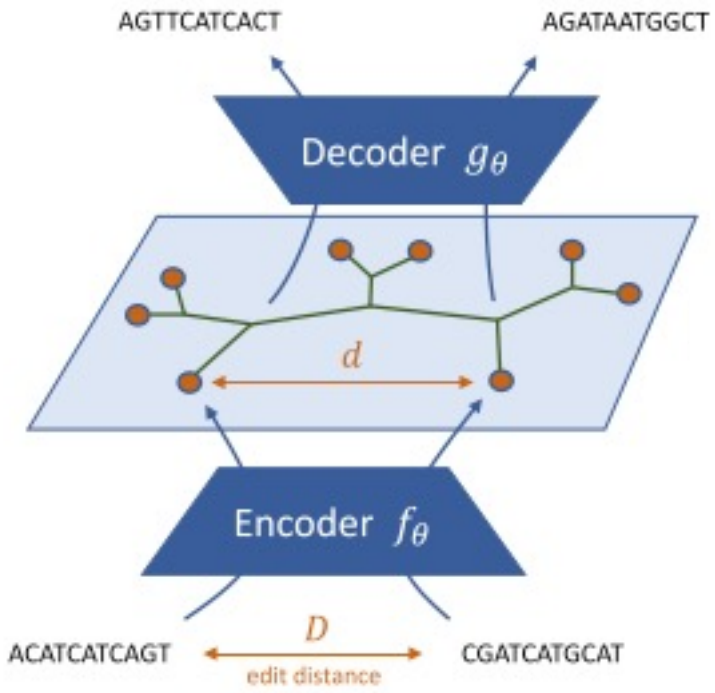
- **Problem of the edit distance computation**
  - Bound to a quadratic complexity and hardly parallelizable
  - Computationally intractable in the accurate computation of similarities among multiple sequences
  - Data independent classical algorithms

    ⇒ Cannot exploit the low-dimensional manifold assumption that characterizes the real world

Proposed approach

- **NeuroSEED** (Neural Distance Embeddings)
  - A general framework to produce representations for biological sequences
  - Learn an encoder function that preserves distances between the sequences and the vector space

    ⇒ Distance in the embedding space is correlated with the evolutionary distance D between sequences



[Main components]

1) Embedding geometry
2) Encoder model
3) Decoder model
4) Loss function

Methods

1. Embedding geometry

- Distance function d between the embedded points

  : Defining the geometry of the embedding space

  - Mostly ignored by previous work, but critical to reflect the relations between the sequences

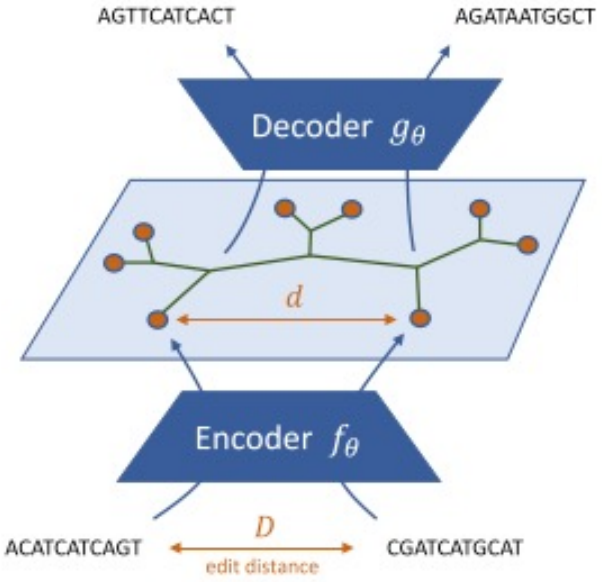
$$\text{Manhattan} \quad d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=0}^{k} |p_i - q_i|$$

$$\text{Euclidean} \quad d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2 = \sqrt{\sum_{i=0}^{k} (p_i - q_i)^2}$$

$$\text{square} \quad d(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2^2 = \sum_{i=0}^{k} (p_i - q_i)^2$$

$$\text{cosine} \quad d(\mathbf{p},\mathbf{q}) = 1 - \frac{\mathbf{p}\cdot\mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|} = 1 - \frac{\sum_{i=0}^{k} p_i q_i}{\sqrt{\sum_{i=0}^{k} p_i^2}\sqrt{\sum_{i=0}^{k} q_i^2}}$$

$$\text{hyperbolic} \quad d(\mathbf{p},\mathbf{q}) = \operatorname{arcosh}\left(1 + 2\frac{\|\mathbf{p}-\mathbf{q}\|^2}{(1-\|\mathbf{p}\|^2)(1-\|\mathbf{q}\|^2)}\right)$$

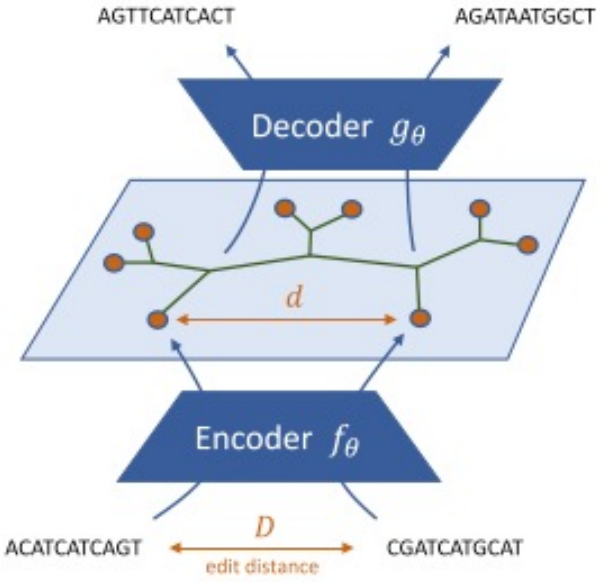


Methods

2. Encoder model

- Mapping sequences to points in the embedding space

- Testing various models as encoder functions
  - Linear layer
  - MLP
  - CNN
  - GRU
  - Transformer with local and global attention
    - Global : every token queries all the others
    - Local : only queries its 2 neighbors



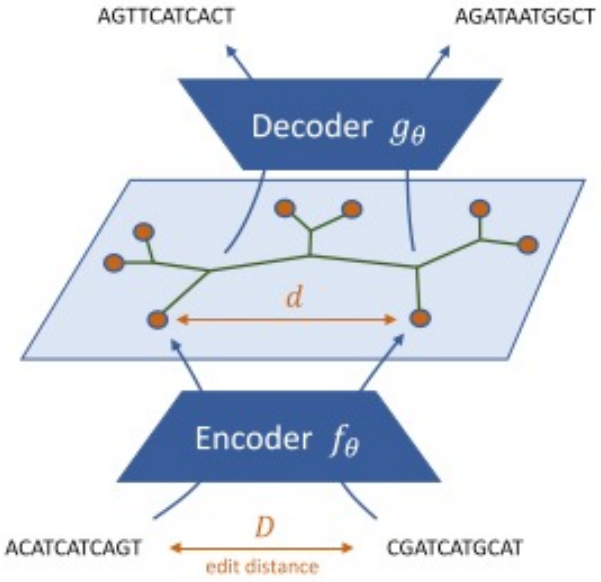
Methods

3. Loss function

- Minimize the mean squared error (MSE) between the sequences' distance and its approximation as the distance between the embeddings

$$L(\theta, S) = \sum_{s_1, s_2 \in S} (D(s_1, s_2) - \alpha\, d(f_\theta(s_1), f_\theta(s_2)))^2$$

- MSE loss can be combined or substituted with other loss functions, depending on the application that the learned embeddings are used for:
  - Closest string retrieval: triplet loss
  - Hierarchical clustering: Dasgupta's cost
  - Multiple sequence loss: sequence reconstruction loss



Methods

4. Decoder

- Training an autoencoder to map sequences to and from a continuous space preserving distances using only pairs of sequence at a time

- **Loss function**

  1) Edit distance approximation component

     : MSE between the real edit distance and a sequence reconstruction

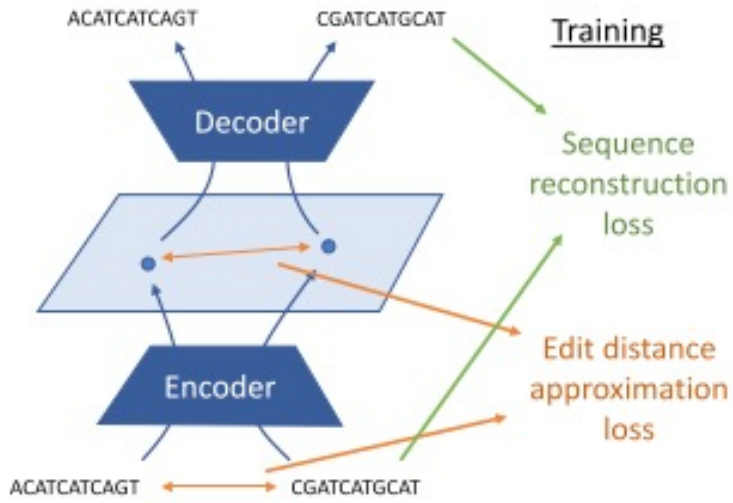
  2) Sequence reconstruction loss

     : mean element-wise cross-entropy loss of the outputs with the real sequences

$$L(\theta, \theta') = \underbrace{(1-\alpha)\, L_{\mathrm{ED}}(\theta)}_{\text{edit distance}} + \underbrace{\alpha\, L_{\mathrm{R}}(\theta, \theta')}_{\text{reconstruction}}$$

$$\text{where} \quad L_{\mathrm{ED}}(\theta) = \left(n^{-1}\, ED(s_1, s_2) - d(f_\theta(s_1), f_\theta(s_2))\right)^2$$

$$\text{and} \quad L_{\mathrm{R}}(\theta, \theta') = \frac{1}{2n} \sum_{i=0}^{n-1} \left(H(s_1[i], g_{\theta'}(f_\theta(s_1))[i]) + H(s_2[i], g_{\theta'}(f_\theta(s_2))[i])\right)$$



Methods

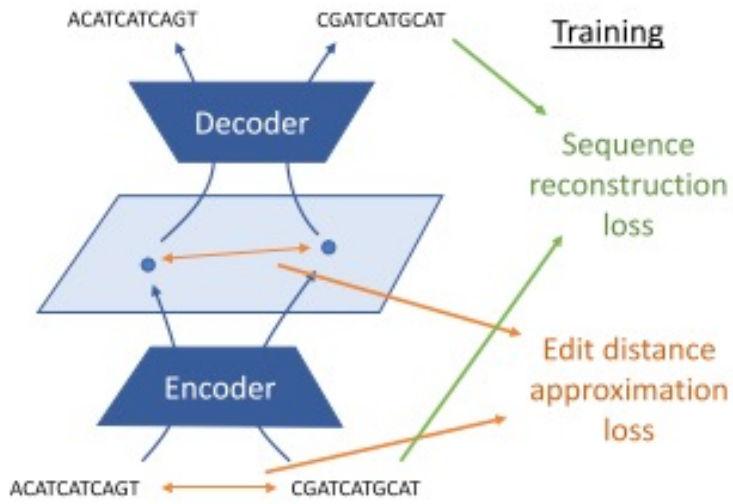
- **Issue**
  - ✓ Only learns the discrete subspace of points to which the generator maps some sequence from the domain
  - ✓ Need to decode points that are outside the subspace hoping to retrieve the string that maps to the point in the subspace closes to it

⇒ During training, adding Gaussian noise to the embedded point in the latent space before decoding **it**

$$L(\theta, \theta') = \underbrace{(1-\alpha)\, L_{\mathrm{ED}}(\theta)}_{\text{edit distance}} + \underbrace{\alpha\, L_{\mathrm{R}}(\theta, \theta')}_{\text{reconstruction}}$$

$$\text{where} \quad L_{\mathrm{ED}}(\theta) = \left(n^{-1}\, ED(s_1, s_2) - d(f_\theta(s_1), f_\theta(s_2))\right)^2$$

$$\text{and} \quad L_{\mathrm{R}}(\theta, \theta', \epsilon) = \frac{1}{2n} \sum_{i=0}^{n-1} \left(H(s_1[i], g_{\theta'}(f_\theta(s_1) + \epsilon_{1i})[i]) + H(s_2[i], g_{\theta'}(f_\theta(s_2) + \epsilon_{2i})[i])\right)$$



Related work

- **Limitation in the previous works**
  1) The lack of analysis of the geometry of the embedding space
  2) The range of tasks is limited to edit distance approximation (EDA) and closest string retrieval (CSR)

[Summary of the previous and the proposed NeuroSEED approaches]

Method	Geometry	Encoder	Decoder	Loss	Tasks
Zheng *et al.* [11]	Jaccard	CNN	✗	MSE	EDA
Chen *et al.* [24]	Cosine	CSM	✗	MSE	EDA
Zhang *et al.* [25]	Euclidean	GRU	✗	MAE + triplet	EDA & CSR
Dai *et al.* [26]	Euclidean	CNN	✗	MAE + triplet	EDA & CSR
Gomez *et al.* [27]	Square	CNN	✗	MSE	EDA & CSR
Section 5	Hyperbolic	CNN & transformer	✗	MSE	EDA
Section 6	Hyperbolic	CNN & transformer	✗	MSE	HC & MSA
Section 7.1	Hyperbolic	Linear	✗	Relaxed Dasgupta	HC
Section 7.2	Cosine	Linear	✓	MSE + reconstr.	MSA
Appendix F	Hyperbolic	CNN & transformer	✗	MSE & triplet	CSR

Experiments

- **Baseline methods**

  1) K-mer
     - Construct a vector where each entry corresponds with the number of occurrences of a k-mer in the sequence
  2) FFP
     - Another alignment-free method which looks at the Jensen-Shannon divergence between distributions of k-mers

- **Three datasets with different portions of the 16S rRNA gene**

Qitta	RT988	Greengenes full-length 16S rRNA DB
More than 6M sequences up to 152 bp covering V4 region	6.7k sequences of length up to 465 bp covering V3-V4 regions	More than 1M sequences between 1,111 to 2.368

Results: Edit distance approximation

- **Metric** : %RMSE $= \frac{100}{n}\sqrt{L(\theta, S)} = \frac{100}{n}\sqrt{\sum_{s_1,s_2 \in S} (ED(s_1, s_2) - n\, d(f_\theta(s_1), f_\theta(s_2)))^2}$ (n: the maximum sequence length)
  - An approximate average error in the distance prediction as a percentage of the size of the sequences

- **Baseline Geometry**

  : Chosen by best average performance

  - K-mers: Cosine
  - FFP: Jensen-Shannon divergence
  - Neural models: Euclidean distance

	RT988		Qiita		Greengenes		
Model	**Baseline**	**Hyperbolic**	**Baseline**	**Hyperbolic**	**Baseline**	**Hyperbolic**	**Training/Inference**
NW alignment	-	-	-	-	-	-	- / 17.5h
4-mer	1.79	-	6.01	-	5.93	-	7s / 7s
5-mer	1.41	-	5.03	-	3.60	-	29s / 29s
6-mer	1.47	-	5.72	-	3.15	-	118s / 118s
FFP 8	12.03	-	20.42	-	10.26	-	360s / 360s
FFP 9	11.86	-	17.53	-	8.63	-	679s / 679s
FFP 10	10.80	-	16.16	-	14.13	-	1274s / 1274s
Linear	21.36±7.07	0.51±0.01	4.39±0.09	2.50±0.01	1155.74±18.34	2.70±0.01	1.1h / 3s
MLP	1.10±0.05	0.59±0.20	4.36±0.19	1.85±0.02	4.38±0.13	2.53±0.03	0.9h / 3s
CNN	0.58±0.05	0.59±0.01	2.68±0.05	**1.56±0.01**	1.37±0.04	**1.00±0.01**	2.1h / 6s
GRU	1.10±0.11	2.56±3.33	3.30±0.06	2.60±0.16	1.61±0.02	1.18±0.16	7.4h / 65s
Global T.	0.52±0.01	**0.46±0.01**	2.10±0.05	1.83±0.03	2.09±0.03	1.91±0.07	2.2h / 3s
Local T.	0.57±0.00	**0.45±0.01**	2.42±0.02	1.86±0.02	1.85±0.04	1.89±0.05	2.0h / 3s

Best

Worst

Fig 2: % RMSE test set results (4 runs)

Results: Edit distance approximation

- **Hyperbolic space**
  - In CNN, the hyperbolic space provides a 22% average RMSE reduction against the best competing geometry for each model
  - It provides significantly more efficient embeddings with the model reaching the elbow at ‘32’ dimension

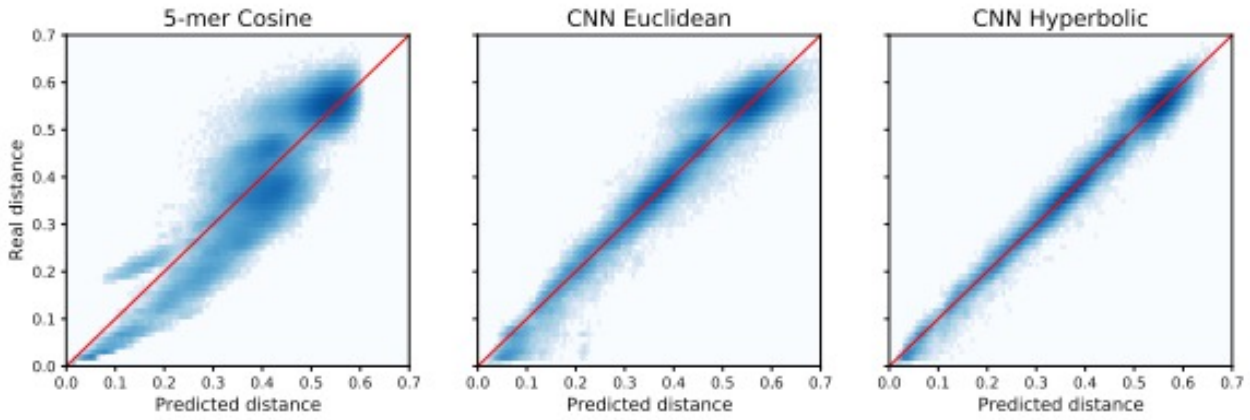


Figure 3: Qualitative comparison in the Qiita dataset between the best performing baseline (5-mer with cosine distance) and the CNN in the Euclidean and hyperbolic space. For every test set sequence pair, predicted vs real distances are plotted, the darkness represents the density of points. The CNN model follows much more tightly the red line of the oracle across the whole range of distances in the hyperbolic space.

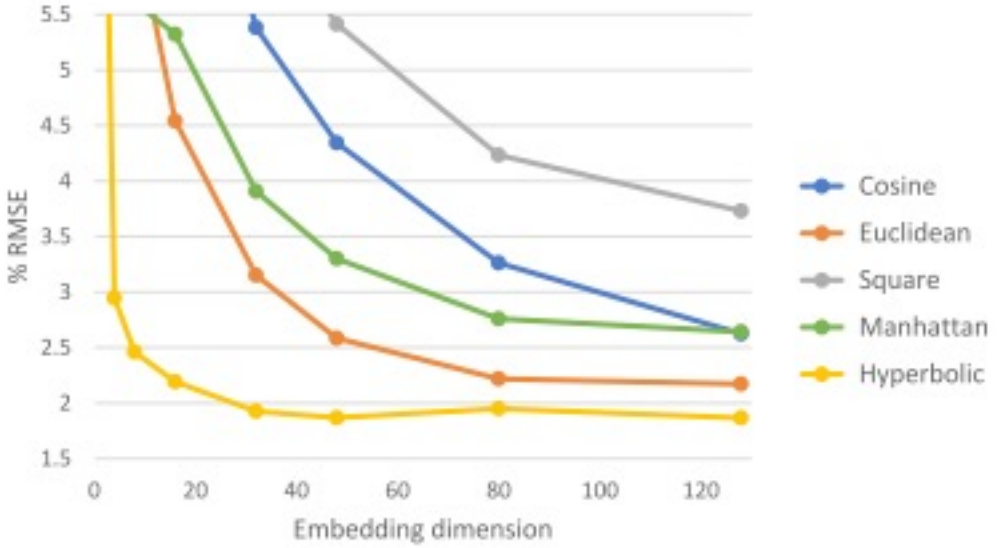


Figure 4: *Edit distance approximation* % RMSE on Qiita dataset for a global transformer with different distance functions.

Results: Unsupervised heuristics

1. Hierarchical clustering (HC)

- Test models with no further tuning from the previous exp on a dataset of 10k unseen sequences from the Qiita
- No statistical difference in the quality of HC produced with ground truth distances
- Significant difference in Dasgupta's cost between different architectures and geometries
- Tree construction time reduced more than 30 minutes

Baselines	
Single L.	0.628
Complete L.	0.479
Average L.	**0.000**

Model	Cosine	Euclidean	Square	Manhattan	Hyperbolic
4-mer	0.261	0.260	0.242	0.191	0.299
Linear	0.062±0.007	0.172±0.036	0.153±0.037	0.177±0.026	0.028±0.005
MLP	0.169±0.054	0.095±0.021	0.289±0.094	0.178±0.029	0.035±0.004
CNN	0.028±0.003	0.030±0.004	0.067±0.022	0.081±0.047	**-0.004±0.015**
GRU	-	0.042±0.006	0.068±0.010	0.069±0.015	0.066±0.043
Global T.	0.032±0.014	**0.003±0.008**	0.038±0.005	**0.002±0.003**	**0.000±0.006**
Local T.	0.035±0.003	0.022±0.008	0.034±0.005	0.022±0.003	**0.000±0.007**

Figure 5: Average Linkage % increase in Dasgupta's cost of NeuroSEED models compared to the performance of clustering on the ground truth distances, ubiquitously used in bioinformatics. Average Linkage was the best performing clustering heuristic across all models.

* Dasgupta's cost: A global objective function that can be associated with the hierarchical clustering trees

Results: Unsupervised heuristics

2. Multiple sequence alignment

- Clustal (The most popular MSA heuristics)
  - Formed by a phylogenetic tree estimation phase that produces a guide tree then used by a progressing alignment phase
  - Hierarchical clustering: bottleneck ⇒ Can be accelerated by NeuroSEED embeddings
- Alignment scores obtained from the attention models are not significantly different from those with the ground truth
- Relatively large variance in the performance

Model	Cosine	Euclidean	Hyperbolic
Linear	60.6±35.1	111.3±3.6	57.5±22.0
MLP	72.3±11.8	53.6±3.1	-11.7±18.9
CNN	31.0±16.2	4.7±9.7	**-16.3±16.1**
Global T.	39.4±74.3	1.9±3.8	31.1±21.8
Local T.	31.9±30.5	8.6±14.1	**-20.1±7.3**

Figure 6: Percentage improvement (average of 3 runs) in the alignment cost (the lower the better) returned by Clustal when using the heuristics to generate the tree as opposed to its default setting using NJ on real distances.

Results: Supervised heuristics with tailored loss functions

1. Hierarchical clustering (HC)

- Use Dasgupta's cost as loss function to embed sequences in the hyperbolic space

$$C_{\text{Dasgupta}}(T; w) = \sum_{ij} w_{ij} \, |\text{leaves}(T[i \vee j])|$$

- Given a rooted binary tree T, for two datapoints i and j, w_{ij} be their pairwise similarity, $i \vee j$ their lowest common ancestor in T and $T[i \vee j]$ the subtree rooted at $i \vee j$
- In this work, $w_j = 1 - d_{ij}$, d_{ij} is the normalized distance between sequence i and j.

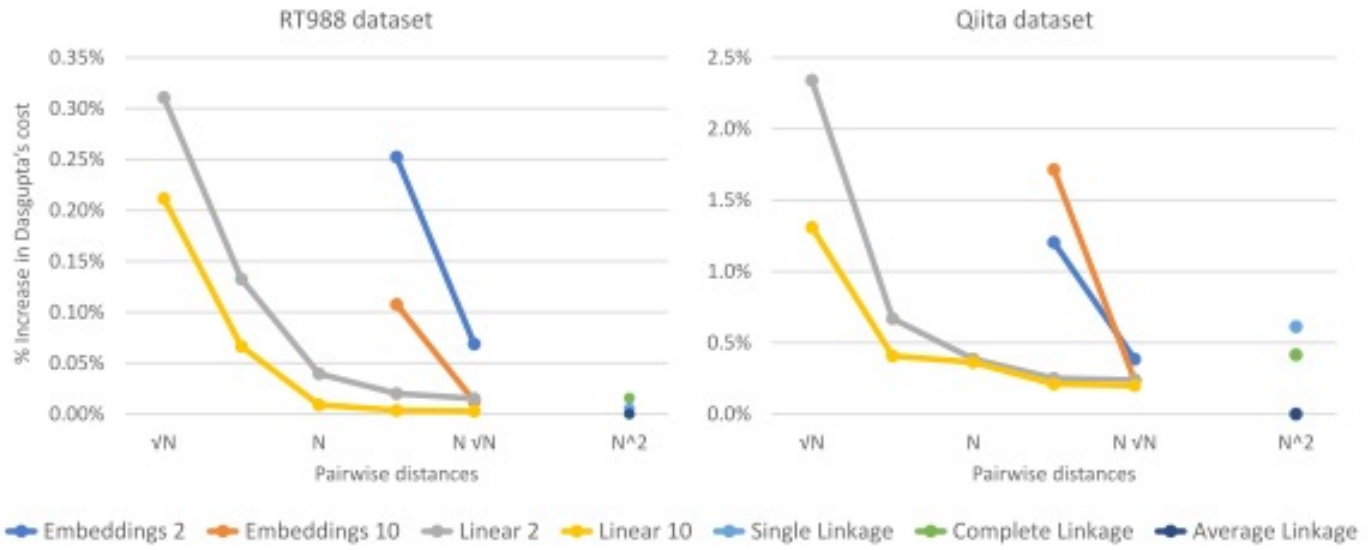
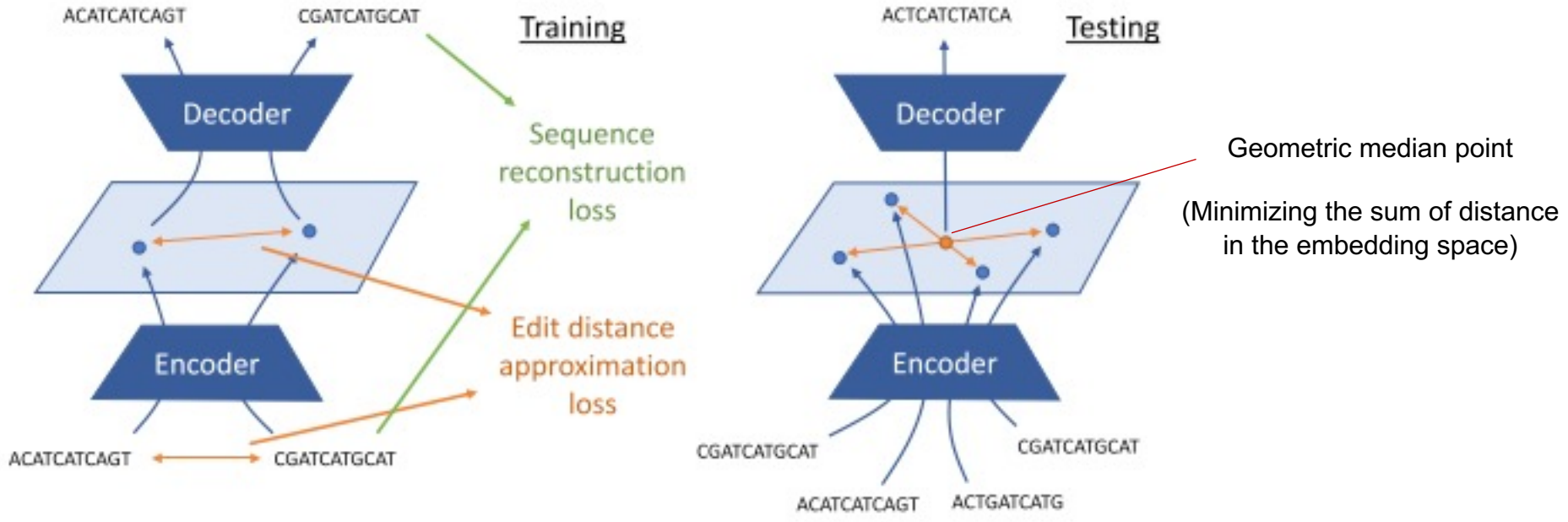


Figure 7: Average Dasgupta's cost of the various approaches with respect to the number of pairwise distances used in the RT988 and Qiita datasets. The performances are reported as the percentage increase in cost compared to the one of the Average Linkage (best performing). Embedding refers to the baseline [20] while Linear to the relaxed `NeuroSEED` approach. The attached number represents the dimension of the hyperbolic space used.

Results: Supervised heuristics with tailored loss functions

2. Steiner string approach to multiple sequence alignment

- Use a decoder from the vector space to convert the Steiner string approximation problem
  1) Training the autoencoder to map sequences to and from a continuous space preserving distances
  2) In testing, the encoder embeds all the sequences in a set, the geometric median point is found, and the decoder is used to find the approximation of the Steiner string



Results: Supervised heuristics with tailored loss functions

- **Metric:** The average consensus error (average distance to the strings in the set)
- **Baselines**
  - Random sequence in the set (Random)
  - The centre string of the set (Centre)
  - Two competitive greedy heuristics proposed by [50] and [51] (greedy-1 and greedy-2)

Baselines	
Random	75.98
Centre	62.52
Greedy-1	**59.43**
Greedy-2	**59.41**

Model	Cosine	Euclidean	Square	Hyperbolic
Linear	**59.41**$\pm$0.11	59.96$\pm$0.27	60.53$\pm$0.49	60.89$\pm$0.82
MLP	60.80$\pm$0.35	60.00$\pm$0.18	59.81$\pm$0.22	59.86$\pm$0.12
CNN	60.96$\pm$0.48	60.20$\pm$0.26	60.76$\pm$1.09	60.48$\pm$0.52

Figure 9: Average consensus error for the baselines (left) and NeuroSEED models (right).

Contribution of this paper

- Introduce a general framework to map sequences in geometric vector spaces
- Show how the hyperbolic space can bring significant improvements to the data-dependent analysis of biological sequences
- Propose several heuristic approaches to classical bioinformatics problems
- Provide significant running time reduction against classical baselines

My thoughts

- Good approach to test different combinations of the neural network model and geometry function
- Somewhat close to the review paper, since this approach is not the first proposed one

- Based on the combination, each dataset shows quite a large difference
  - Need to perform multiple experiments
  - Require time to find the optimal one, which would be much more than just using the original distance
  - Hard to use this framework in the practical manner

- Utilizing hyperbolic space only shows better performance for some task
  - Especially on the edit distance approximation
  - Depends on the neural network model

- Little bit hard to follow the manuscript as the paper was not organized in smoothly