Levenshtein Distance Embedding with Poisson Regression for DNA Storage

Xiang Wei[*1], **Alan J.X. Guo**[*†1], **Sihan Sun**[1], **Mengyi Wei**[1], **Wei Yu**[2]

[1] Center for Applied Mathematics, Tianjin University, No. 92, Weijin Road, Tianjin, 300072, China
[2] China Mobile Research Institute, No. 32, Xuanwumen West Street, Beijing, 100053, China
{weixiang, jiaxiang.guo, sihansun, mengyi.wei}@tju.edu.cn, YuweiNKU@outlook.com

Abstract

Efficient computation or approximation of Levenshtein distance, a widely-used metric for evaluating sequence similarity, has attracted significant attention with the emergence of DNA storage and other biological applications. Sequence embedding, which maps Levenshtein distance to a conventional distance between embedding vectors, has emerged as a promising solution. In this paper, a novel neural network-based sequence embedding technique using Poisson regression is proposed. We first provide a theoretical analysis of the impact of embedding dimension on model performance and present a criterion for selecting an appropriate embedding dimension. Under this embedding dimension, the Poisson regression is introduced by assuming the Levenshtein distance between sequences of fixed length following a Poisson distribution, which naturally aligns with the definition of Levenshtein distance. Moreover, from the perspective of the distribution of embedding distances, Poisson regression approximates the negative log likelihood of the chi-squared distribution and offers advancements in removing the skewness. Through comprehensive experiments on real DNA storage data, we demonstrate the superior performance of the proposed method compared to state-of-the-art approaches.

Introduction

The Levenshtein distance (Levenshtein et al. 1966) (also known as the edit distance) between two sequences is defined as the minimum number of insertions, deletions, or substitutions required to modify one sequence into another. The dynamic programming algorithm introduced in (Wagner and Fischer 1974) is commonly employed for accurate calculation of the Levenshtein distance. However, this method has a computational complexity of $O(mn)$ for two strings of length m and n. According to Theorem 1 from (Backurs and Indyk 2015),

Theorem 1. *Given two sequences of length n, the Levenshtein distance can't be computed in time $O(n^{2-\delta}), \forall\, \delta > 0$, otherwise the Strong Exponential Time Hypothesis would be violated.*

[*]These authors contributed equally.
[†]Corresponding Author.

a method of calculating Levenshtein distance in linear complexity is not reachable. The computational complexity of the Levenshtein distance poses limitations on its application at a large scale, particularly in the field of DNA storage (Rashtchian et al. 2017; Dong et al. 2020).

The applications of the Levenshtein distance encompass a wide range of domains, including optical character recognition (Haldar and Mukhopadhyay 2011), text plagiarism detection (Su et al. 2008), entity linking (Jiang et al. 2014), *etc.* In the field of bioinformatics, where nucleic acids and proteins are represented as sequences of their basic building blocks, the Levenshtein distance finds widespread applications. It is utilized in tasks such as multiple sequence alignment (Li and Homer 2010), biological database retrieval (Berger, Waterman, and Yu 2020), sequence hierarchical clustering (Sberro et al. 2019), *etc.* In recent years, the rapid development of DNA storage (Goldman et al. 2013; Church, Gao, and Kosuri 2012; Grass et al. 2015) has introduced numerous applications of the Levenshtein distance, including sequence clustering (Rashtchian et al. 2017; Zorita, Cuscó, and Filion 2015; Qu, Yan, and Wu 2022; Logan et al. 2022), sequence alignment (Li and Homer 2010; Corso et al. 2021), synchronization channel coding (Press et al. 2020; Bar-Lev, Etzion, and Yaakobi 2023; Welzel et al. 2023), *etc.* However, as the scale of information stored by DNA molecules continues to grow, the computational complexity of the Levenshtein distance becomes a significant challenge for the aforementioned applications.

To address the challenge of high computational complexity, various methods have been proposed to approximate the Levenshtein distance. For example, in (Ostrovsky and Rabani 2007), the authors utilized ℓ_1 distance to approximate the Levenshtein distance of $0, 1$-words with low distortion. Another approach, the CGK algorithm (Chakraborty, Goldenberg, and Koucký 2016), maps the Levenshtein distance to Hamming distance through a randomized injective embedding. Both of these methods use low-complexity distances as approximations for the Levenshtein distance.

Neural network-based methods have also been explored. The gated recurrent unit (GRU) (Cho et al. 2014) has been adopted to embed the Levenshtein distance into Euclidean space (Zhang, Yuan, and Indyk 2019), preserving the relative order between the sequences. In (Dai et al. 2020), a CNN-based embedding pipeline for the Levenshtein dis-

tance was proposed. The authors provided theoretical evidence for employing a convolutional neural network (CNN) as the embedding model. In (Corso et al. 2021), a framework for embedding biological sequences in geometric vector spaces was proposed. In their work, the DNA sequences are embedded into hyperbolic space instead of Euclidean space to capture the implicit hierarchical structure from biological relations between the sequences (Chami et al. 2020). Additionally, squared Euclidean distance has also been used to embed the Levenshtein distance (Guo, Liang, and Hou 2022). This work established connections between the Levenshtein distance and degree of freedom with the output distribution of the embedding model, and proposed the so-called chi-square regression.

In this paper, a neural network-based Levenshtein distance embedding algorithm is proposed with the Poisson regression. We first demonstrate that the choice of embedding dimension has a theoretical impact on the distribution of the approximation, which in turn affects the approximation precision. Leveraging these theoretical analyses, we introduce a method to determine the appropriate embedding dimension for a given dataset and embedding network. Once the embedding dimension is determined, we employ Poisson regression to train the embedding model. The Poisson regression offers two significant advantages. Firstly, it naturally aligns with the definition of Levenshtein distance. When the Levenshtein distance is small, it can be approximately interpreted as counting the event of edit operations from a fixed interval. Secondly, it approximates the negative log likelihood of the chi-squared distribution and shows advances in removing the skewness. Experimental results in real-world scenarios validate the efficacy of the theoretical analysis on the choice of embedding dimension. Through experiments conducted on DNA storage data, it is demonstrated that the proposed method outperforms state-of-the-art approaches.

Framework of the Embedding Method

In this section, we provide a brief overview of the embedding method framework and highlight some of the assumptions that have been previously validated in (Guo, Liang, and Hou 2022).

Most of the sequence embedding methods explictly (Guo, Liang, and Hou 2022; Zheng et al. 2019) or implicitly (Cho et al. 2014; Dai et al. 2020) adopt the framework of Siamese neural network (Bromley et al. 1993; He et al. 2020). In this framework, the embedding network $f(\cdot;\theta)$ maps a sequence $\boldsymbol{s}$ to its corresponding embedding vector $\boldsymbol{u} = f(\boldsymbol{s};\theta)$ using parameter θ. The embedding network $f(\cdot;\theta)$ is optimized by training two identical branches in the Siamese neural network. Let $((\boldsymbol{s},\boldsymbol{t}),d)$ be a sample from the training data, where the $\boldsymbol{s}$ and $\boldsymbol{t}$ are two sequences, and d represents the groundtruth Levenshtein distance between them. The parameter θ is trained by minimizing the difference between the groundtruth Levenshtein distance and the approximation, as shown in the following equation:

$$\begin{aligned}\hat{\theta} &= \arg\min_{\theta} \mathcal{L}(\boldsymbol{d}, \hat{\boldsymbol{d}}; \theta) \\ &= \arg\min_{\theta} \sum \mathcal{L}(d, \Delta(f(\boldsymbol{s};\theta), f(\boldsymbol{t};\theta))).\end{aligned} \tag{1}$$

Here, the $\mathcal{L}$ is a predefined loss function, and the $\Delta(\boldsymbol{u},\boldsymbol{v})$ denotes the distance between embedding vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ in the embedding space. During the testing phase, the sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ are mapped to their respective embedding vectors $\boldsymbol{u} = f(\boldsymbol{s};\hat{\theta})$ and $\boldsymbol{v} = f(\boldsymbol{t};\hat{\theta})$ with the optimized parameter $\hat{\theta}$. The distance $\Delta(\boldsymbol{u},\boldsymbol{v})$ between the embedding vectors is then used as an approximation of the Levenshtein distance between the sequences $\boldsymbol{s}$ and $\boldsymbol{t}$.

In previous work, it has been shown that the squared Euclidean distance, defined as:

$$d_{\ell_2^2}(\boldsymbol{u},\boldsymbol{v}) = ||\boldsymbol{u}-\boldsymbol{v}||_2^2 = \sum_i (u_i - v_i)^2, \tag{2}$$

can effectively approximate the Levenshtein distance (Guo, Liang, and Hou 2022). Although the squared Euclidean distance is not a true metric, the application of squared Euclidean distance not only offers good approximation precision, but also establishes a connection between the Levenshtein distance and the degree of freedom with the difference of the embedding vectors. Specifically, under certain assumptions, it was derived that for pairs of sequences with a consistent Levenshtein distance d, the distribution of the embedding vector $\boldsymbol{u}-\boldsymbol{v}$ has a degree d of freedom. To align with their work, we briefly describe these assumptions as follows:

- A1 By deploying a batch normalization layer, each element u_i of the embedding vector $\boldsymbol{u}$ follows the standard normal distribution $N(0,1)$;
- A2 The embedding vector $\boldsymbol{u}$ has no information redundancy. In other words, if $i \neq j$, the embedding elements u_i and u_j of $\boldsymbol{u}$ are independent of each other;
- A3 Embedding vectors of non-related sequences are independent of each other.

These assumptions allow for theoretical analysis and facilitate the connection between the Levenshtein distance and the squared Euclidean distance.

Embedding Dimension and Approximation Precision

Why Large Embedding Dimension Improves the Approximation Precision?

The average of the groundtruth Levenshtein distance between the independent sequences is a statistic feature of the dataset. A good embedding method should ensure that the expectation of predicted distance between independent embedding vectors matches the average of groundtruth Levenshtein distance on pairs of independent sequences. To achieve this, a scale factor can be utilized to help the model satisfy this requirement. Let $\boldsymbol{s}$ and $\boldsymbol{t}$ be two independent sequences from a fixed dataset, and $\boldsymbol{u} = f(\boldsymbol{s})$ and $\boldsymbol{v} = f(\boldsymbol{t})$

be their corresponding embedding vectors of length n. The embedding method (Guo, Liang, and Hou 2022)empirically selects 80 as the length of embedding vectors, and the corresponding scale factor is set to be $\sqrt{2}/2$. To adapt this framework to arbitrary dimension n, we use a scale factor $r(n)$ with respect to the embedding dimension n to calculate the approximated distance as follows:

$$\hat{d}(\boldsymbol{s},\boldsymbol{t}) = d_{\ell_2^2}(r(n)\boldsymbol{u}, r(n)\boldsymbol{v}) = r^2(n)\sum_{i=1}^{n}(u_i - v_i)^2. \quad (3)$$

Based on the three assumptions mentioned earlier, where u_i and v_i independently follow $N(0,1)$, it can be inferred that $u_i - v_i$ follows $N(0,2)$, and $\frac{1}{2}(u_i - v_i)^2$ follows $\chi^2(1)$. Therefore, the approximated distance between the independent sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ follows a chi-squared distribution with n degree of freedom

$$\frac{1}{2r^2(n)}\hat{d}(\boldsymbol{s},\boldsymbol{t}) \sim \chi^2(n). \quad (4)$$

The expectation of $\hat{d}(\boldsymbol{s},\boldsymbol{t})$ can be easily calculated as:

$$\mathbb{E}[\hat{d}(\boldsymbol{s},\boldsymbol{t})] = 2nr^2(n). \quad (5)$$

Let M be the average Levenshtein distance between independent sequences over the specific dataset. In order to satisfy Equation (5), the scale factor can be set as:

$$r(n) = \sqrt{\frac{M}{2n}}. \quad (6)$$

Thus ensuring that the expected distance between independent embedding vectors aligns with the average Levenshtein distance between independent sequences in the dataset.

An embedding method primarily focuses on approximating the Levenshtein distance between correlated sequences. We will now demonstrate how the embedding dimension n affects the distribution of the approximation for such pairs of sequences. Consider two correlated sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ with a groundtruth Levenshtein distance of d. Using a slight abuse of notation, let $\tilde{\boldsymbol{u}} = r(n)\boldsymbol{u}$ and $\tilde{\boldsymbol{v}} = r(n)\boldsymbol{v}$ denote the respective scaled embedding vectors. It is assumed that the difference $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$ has a degree of freedom proportional to the groundtruth distance d, which can be expressed as follows:

$$\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}} = \boldsymbol{y}\boldsymbol{P} = (y_1, y_2..., y_m, 0..., 0)\sqrt{\frac{M}{n}}\boldsymbol{P} \quad (7)$$

where the non-zero elements y_i of $\boldsymbol{y}$ are independently and identically distributed (i.i.d.) random variables following the standard normal distribution $N(0,1)$, the matrix $\boldsymbol{P}$ is an orthogonal matrix, and m is called the degree of freedom with $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$ and equals nd/M. Note that under assumptions A2 and A3, when $d = M$, we have m equal to the embedding dimension n. Although m is rarely an integer, Equation (7) is considered acceptable if the model can learn to approximate integers close to m.

Now, let's calculate the squared Euclidean distance between the embedding vectors $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$:

$$\begin{aligned} d_{l_2^2}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) &= (\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}})(\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}})^T = \frac{M}{n} y\boldsymbol{P}\boldsymbol{P}^T y^T \\ &= \frac{M}{n} yy^T = \frac{M}{n}\sum_{i=1}^{m} y_i^2. \end{aligned} \quad (8)$$

This formula shows that the distribution of the approximated distance $d_{l_2^2}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ for a correlated pair of sequences with Levenshtein distance d follows the chi-squared distribution:

$$\frac{n}{M} d_{l_2^2}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) \sim \chi^2(m) = \chi^2(\frac{n}{M}d). \quad (9)$$

It can be observed that the expected value of the approximated distribution for correlated pairs is equal to their Levenshtein distance, which serves as the learning target for the model. In order to simplify the notations, let's introduce a new variable $k = n/M$. Consequently, the Equation (9) is rewrited as

$$k d_{l_2^2}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) \sim \chi^2(kd), \quad k = \frac{n}{M}. \quad (10)$$

Given a sequence pair $(\boldsymbol{s},\boldsymbol{t})$ with a groundtruth Levenshtein distance d, it is evident that the precision of the approximation is affected by the variance in the distribution. Referring to Equation (10), the variance of approximations can be calculated as:

$$\mathrm{Var}[d_{l_2^2}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})] = \frac{2d}{k} = \frac{2dM}{n}. \quad (11)$$

When an embedding network $f(\cdot, \hat{\theta})$ is well-trained, one can deduce from Equation (11) that a larger embedding dimension n leads to smaller variance in the approximations, resulting in higher precision. It is worth noting that, based on Equation (11), regardless of the choice of k or the embedding dimension n, sequences with smaller Levenshtein distance exhibit higher approximation precision.

Deciding the Appropriate Embedding Dimension

The embedding dimension n is a critical hyperparameter. As we discussed previously, there exists a relationship between the variance of approximated distances and the embedding dimension, indicating that a larger n leads to a decrease in variance and higher theoretical approximation precision for the learned model. Therefore, the learned model has higher theoretical approximation precision with a larger n. However, it is important to note that this assertion may not hold universally due to a potential violation of assumption A2. Referring to Equation (10), the degree of freedom with $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$ is nd/M. Consequently, increasing the embedding dimension n results in a higher degree of freedom with $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$. For a pair of sequences from a specific dataset, the information capacity is limited since these sequences are discrete and possess finite lengths. Hence, in the context of non-generative neural networks, the degree of freedom with the difference between the embedding vectors $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$ does not reach infinity. Additionally, different network architectures possess varying abilities in feature extraction, which can further restrict the expressiveness of the degree of freedom. Therefore, the attainable degree of freedom may be constrained by both the dataset's inherent information limitations and the specific network architecture employed. In summary, we propose the following assumption as a supplementary

A4 On a fixed dataset and network architecture, there is an upper bound n_0 on the embedding dimension n. When $n \leq n_0$, it is possible to achieve full freedom (n degrees

of freedom) in the embedding $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$, when $n > n_0$, the embedding $\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}$ can have at most n_0 degrees of freedom.

The relationship between the embedding dimension n and the variance of approximations in Equation (11) suggests that a larger embedding dimension can potentially improve the theoretical approximation precision, while according to the assumption A4, an embedding dimension beyond n_0 would be unnecessary and may even hinder the method's performance. Therefore, it is worth to determine the appropriate upper bound n_0 and utilize it as the embedding dimension.

Embedding Dimension Searching

To determine the appropriate embedding dimension n_0, one could start by considering assumption A4 and Equation (7). Given an embedding dimension n and a fully trained embedding network $f(\cdot;\hat{\theta})$, collect a set of mutually non-related sequences $\{\boldsymbol{s}_i\}$ along with their corresponding embedding vectors $\{\boldsymbol{u}_i\} = f(\boldsymbol{s}_i;\hat{\theta})$. According to assumption A4, if $n \leq n_0$, the difference between the embedding vectors can be expressed as a linear combination of n independent standard normal variables, regarding the scale factor. This can be expressed as:

$$\boldsymbol{u_i} - \boldsymbol{u_j} = (y_1, y_2, \ldots, y_n)\boldsymbol{P} \quad (i \neq j), \tag{12}$$

where y_is are i.i.d. and follow the standard normal distribution $N(0,1)$, and the matrix $\boldsymbol{P}$ is an orthogonal matrix. On the other hand, when $n > n_0$, the expression becomes:

$$\boldsymbol{u_i} - \boldsymbol{u_j} = (y_1, y_2, \ldots, y_{n_0}, 0, \ldots, 0)\boldsymbol{P} \quad (i \neq j). \tag{13}$$

This leads to consider that the spectral decomposition of the covariance matrix $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j})$, which is also a key step in naive principal component analysis (PCA). When $n \leq n_0$, the eigenvalues of the covariance matrix are around 1, when $n > n_0$, the sorted eigenvalues exhibit a steep decline, reaching close to zero. Therefore, by gradually increasing the embedding dimension and monitoring the decreasing pattern of the sorted eigenvalues, we can determine the appropriate embedding dimension n_0. We refer to this value of n_0 as the early-stopping dimension (ESD), and adopt it as our choice for the suitable embedding dimension.

Poisson Regression

In previous studies on Levenshtein distance embedding, the mean square error (MSE), the mean absolute error (MAE), or a combination of both losses have often been employed to optimize the model. However, these losses penalize the predicted distance $\hat{d}$ symmetrically with respect to its groundtruth d. This disregards the fact that the distribution of the predicted distance $\hat{d}$ exhibits a clear bias when d is small. In (Guo, Liang, and Hou 2022), they introduced the so-called chi-squared regression to address this issue. The negative log-likelihood loss of chi-squared distribution

$$\mathrm{RE}\chi^2(\hat{d}, d) = \hat{d} - (d-2)\ln\hat{d} \tag{14}$$

is used as their loss function. It can be observed from Equation (14) that this loss function is minimized at $\hat{d} = d - 2$ rather than $\hat{d} = d$. To overcome the drawbacks of these existing losses, we propose the adoption of Poisson regression from two different perspectives.

Poisson Regression: Interpreting Levenshtein Distance with Poisson Distribution

The Poisson distribution represents the distribution of events occurring within a fixed interval in constant mean rate. The probability mass function of Poisson distribution with expectation λ is:

$$P(k;\lambda) = \frac{\lambda^k \mathrm{e}^{-\lambda}}{k!}, \quad k = 0, 1, 2, \ldots. \tag{15}$$

In the context of Levenshtein distance, which measures the minimum number of deletions, insertions, and substitutions needed to transform one sequence into another, we can approximate the Levenshtein distance as a Poisson-distributed random variable when the distance is significantly smaller than the length of the sequences. This approximation treats the sequence as a fixed interval and the operations as random events occurring within this interval. In Poisson regression, the model parameter θ is estimated based on the maximum likelihood of Poisson distribution. The optimization target is typically the negative log-likelihood loss with the Poisson distribution (PNLL) of the groundtruth:

$$\mathrm{PNLL}(\hat{d}, d) = \hat{d} - d\ln\hat{d} + \ln d!. \tag{16}$$

Omitting the terms that that do not contribute to the gradients, the PNLL loss can be rewritten as:

$$\mathrm{PNLL}(\hat{d}, d) \doteq \hat{d} - d\ln\hat{d}. \tag{17}$$

In Poisson regression, the loss defined in Equation (17) reaches its minimum value when $\hat{d} = d$, and also derives asymmetric penalization on $\hat{d}$ around its groundtruth value d.

Poisson Regression: an Asymptotic Chi-Squared Regression

Recall that the approximation of Levenshtein distance theoretically follows a chi-squared distribution in Equation (10) under the discussed assumptions. In contrast, the proposed Poisson regression utilizes a Poisson distribution, which differs from the chi-squared distribution in Equation (10). This raises concerns about the potential impact on the performance of the proposed method. However, the following discussion aims to address and alleviate such doubts by providing another explaination on Poisson regression.

The PNLL in Equation (17) can be viewed as an approximation of the negative log-likelihood of distribution in Equation (10) when k is sufficiently large. Recall that the distribution of approximations follows Equation (10) with a parameter k. Note that the probability density function of $\chi^2(d)$ is

$$p(x;d) = (2^{\frac{d}{2}}\Gamma\left(d/2\right))^{-1} x^{\frac{d}{2}-1}\mathrm{e}^{-\frac{x}{2}} \tag{18}$$

where $\Gamma(\cdot)$ is the Gamma function. One step further, the probability density function $p(x; k, d)$ of Equation (10) is calculated as

$$p(x; k, d) = \frac{k}{2^{\frac{kd}{2}}\Gamma\left(\frac{kd}{2}\right)}(kx)^{\frac{kd}{2}-1}\mathrm{e}^{-\frac{kx}{2}}. \tag{19}$$

By Equation (19), we can compute the negative log-likelihood of a predicted $\hat{d}$ from the chi-squared distribution in Equation (10) as

$$\begin{aligned}-\ln p(\hat{d}; k, d) =& \frac{k\hat{d}}{2} - (\frac{kd}{2} - 1)\ln(k\hat{d}) \\ & - \ln k + \frac{kd}{2}\ln 2 + \ln\Gamma\left(\frac{kd}{2}\right).\end{aligned} \tag{20}$$

By omitting terms that contribute zero to the gradients and rescaling the equation by multiplying $2/k$, the negative log-likelihood loss is

$$-\ln p(\hat{d}; k, d) \doteq \hat{d} - (d - \frac{2}{k})\ln\hat{d}. \tag{21}$$

It is evident that when $k \to +\infty$, the PNLL in Equation (17) is the limit of Equation (21). This observation suggests that the PNLL is an approximation of the negative log-likelihood of the chi-squared distribution in Equation (10). It is worth noting that as k increases, the Equation (21) converges to Equation (17), and the minimum point of the loss in Equation (21) converges to the groundtruth d.

As mentioned above, the chosen embedding dimension n (proportional to k) is selected to be as large as possible within the limit of n_0. Considering the above discussion, the choice of Poisson regression preserves the advantage of chi-squared regression in terms of asymmetric penalization and addresses the concern of skewness in the chi-squared distribution.

Experiments

Dataset The experiments are conducted using the DNA storage data introduced in (Guo, Liang, and Hou 2022). This data[1] was originally provided by a research called DNA-Fountain (Erlich and Zielinski 2017), which is a milestone in DNA storage. Each data sample from the DNA storage dataset can be represented as a tuple $((\boldsymbol{s}, \boldsymbol{t}), d)$, where $(\boldsymbol{s}, \boldsymbol{t})$ is a pair of DNA sequences and d is the groundtruth Levenshtein distance between these sequences. The DNA sequences in the dataset are strings composed of oligonucleotides and sequencing failure bases, denoted by the alphabet $\{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}, \mathrm{N}\}$. The samples can be categorized into two classes of homologous and non-homologous samples. In a homologous sample $((\boldsymbol{s}, \boldsymbol{t}), d)$, the sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ belong to the same cluster obtained from the DNA-storage pipeline and have a small Levenshtein distance d. On the other hand, in a non-homologous sample $((\boldsymbol{s}, \boldsymbol{t}), d)$, the sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ belong to different clusters and can be considered as independent sequences with a large Levenshtein distance d. The length of the DNA sequences in the dataset is approximately 152, and they are padded to a fixed length of 160 before being fed into the embedding network. To ensure the separation of data for training and testing, the training and testing sets of the DNA storage data are divided by a partition on the clusters of the retrieved DNA sequences.

Metric Two metrics are used to evaluate the performance of the model, as they are, the global approximation error (AE_g) and the homologous approximation error (AE_h). The AE_g is calculated as the mean absolute error over the testing set Te

$$\mathrm{AE}_g = \frac{1}{\#\mathrm{Te}} \sum_{((\boldsymbol{s},\boldsymbol{t}),d)\in\mathrm{Te}} |\hat{d} - d|, \tag{22}$$

while the AE_h is the mean absolute error over the homologous samples Te_h from the testing set

$$\mathrm{AE}_h = \frac{1}{\#\mathrm{Te}_h} \sum_{((\boldsymbol{s},\boldsymbol{t}),d)\in\mathrm{Te}_h} |d - \hat{d}|, \tag{23}$$

where the $\#\mathrm{Te}$ and $\#\mathrm{Te}_h$ are the number of samples in Te and Te_h, respectively. The values of AE_g and AE_h provide insights into the overall approximation error and the specific performance on homologous samples, respectively. In general, the Levenshtein distance between two completely unrelated sequences is considered of lesser importance. Therefore, the AE_h is a more important metric than AE_g in this study.

Models In the experiments, we employ a variety of neural network structures as the embedding network, including the commonly used CNN-5, CNN-10, and GRU networks as utilized in previous studies (Zhang, Yuan, and Indyk 2019; Dai et al. 2020; Guo, Liang, and Hou 2022). Additionally, to explore the impact of network depth and width on the ESD and model performance, we also use wider embedding networks, denoted as CNN-5-w and CNN-10-w. The only difference between CNN-$*$-w and CNN-$*$ is that the CNN-$*$-w has more convolution channels. These models are the 1D versions from their original model. Further details can be found in the Appendices[2].

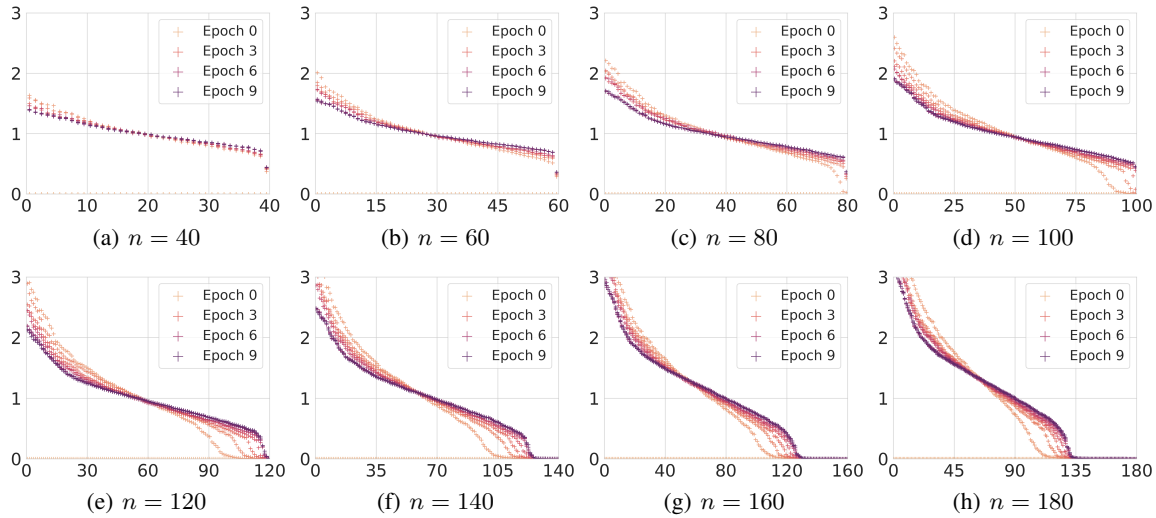
The Early-Stopping Dimension

It has been analyzed that there exists an appropriate embedding dimension called the ESD n_0, with the fixed dataset and network architecture. The theoretical approximation precision improves as the embedding dimension n increases, up to the point where n reaches the ESD n_0.

Specifically, there exists the ESD n_0. When the embedding dimension n is less than or equal to n_0, the eigenvalues of $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$ are approximately equal to 1. When n exceeds n_0, the sorted eigenvalues gradually decrease to 0 after reaching n_0. To verify this phenomenon, we train embedding networks with different embedding dimensions, and plot the sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$. The results for the embedding network of CNN-5 are plotted in Figure 1. It is

[1]The data can be accessed through https://github.com/TeamErlich/dna-fountain and https://www.ebi.ac.uk/ena/data/view/PRJEB19305.

[2]Please refer to https://arxiv.org/abs/2312.07931 for the Appendices.



(a) $n = 40$ (b) $n = 60$ (c) $n = 80$ (d) $n = 100$

(e) $n = 120$ (f) $n = 140$ (g) $n = 160$ (h) $n = 180$

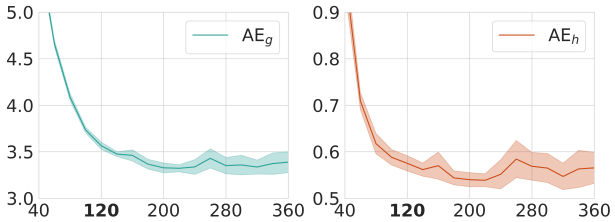
Figure 1: The sorted eigenvalues of $\text{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$ are plotted for different choices of the embedding dimension n in the CNN-5 embedding network. When n is small, the eigenvalues are distributed around 1, as in (a)–(d). Increasing the embedding dimension n, the sorted eigenvalues decrease to 0 after some dimension, as in (e)–(h).

suggested that, when the embedding dimension n is small, the eigenvalues are distributed around 1, indicating that n is smaller than the ESD n_0. As the embedding dimension increases, the sorted eigenvalues sharply decrease to 0 after a certain dimension. And regardless of how large the embedding dimension n is, the number of non-zero eigenvalues remains relatively constant. This observation suggests that the embedding dimension exceeds the value of n_0. Based on the results shown in Figure 1, we determine that the ESD n_0 for the engaged dataset and the CNN-5 is approximately $n_0 = 120$.

It is worth noting that, when using the CNN-5-w or CNN-10-w as the embedding network, the phenomenon of sudden vanishing eigenvalues along increasing embedding dimension becomes more evident. Furthermore, the curves of approximation errors are more stable on the embedding networks CNN-5-w and CNN-10-w. For more details, please refer to the Appendices.

To verify the claim that the performance improves as the embedding dimension n increases until reaching the ESD n_0, the approximation precision of the CNN-5 model with different embedding dimension is plotted in Figure 2. This figure suggests that increasing the embedding dimension leads to a decrease in the approximation error when the embedding dimension n is less than the ESD $n_0 = 120$. When the embedding dimension n exceeds the ESD $n_0 = 120$, the improvement in approximation precision becomes negligible, and the standard deviation of the approximation error across different runs grows. This indicates that the stability of the embedding network in training phase is compromised.

In the Appendices, we provide similar figures to those in Figure 1 and Figure 2 for the CNN-10, CNN-5-w, CNN-10-w, and GRU embedding networks used in this study. The ESD values corresponding to these four networks are CNN-10: $n_0 = 120$, CNN-5-w: $n_0 = 140$, CNN-10-w: $n_0 = 140$,



(a) AE$_g$ (b) AE$_h$

Figure 2: The global approximation error and homologous approximation error are shown against the embedding dimension in (a) and (b), respectively. The curves are plotted based on the mean and standard deviation over 5 runs. The approximation errors decrease alongwith increase the embedding dimension n until the ESD n_0, which is 120 for CNN-5. When the $n > n_0$, there is no gain of the performance on a larger n, but the model performance becomes unstable with a larger standard deviation.

and GRU: $n_0 = 140$. Considering those figures and the results in the comparative experiments, we may speculate that the width of the network plays a more significant role than the depth of the network in improving the approximation precision.

The Comparative Experiments

We conduct experiments using different combinations of embedding networks, loss functions, and embedding dimensions. The embedding networks include CNN-5, CNN-10, CNN-5-w, CNN-10-w, and GRU. The loss functions used in the experiments are MSE, MAE, REχ^2 from (Guo, Liang, and Hou 2022), and the proposed PNLL. The choices of embedding dimensions are an empirical fixed dimension of

Embedding Network	Metric	Objective Function			
		MSE	MAE	REχ^2	PNLL
CNN-5(80)	AE_g	4.06±0.07	**4.00±0.02**	4.32±0.01	4.09±0.04
	AE_h	0.83±0.02	0.68±0.02	0.85±0.00	**0.62±0.02**
CNN-5(120)	AE_g	**3.48±0.02**	3.52±0.07	4.15±0.07	3.57±0.04
	AE_h	0.82±0.02	0.64±0.03	0.83±0.01	**0.58±0.02**
CNN-10(80)	AE_g	3.78±0.02	**3.76±0.08**	4.24±0.05	3.90±0.02
	AE_h	0.72±0.02	0.61±0.01	0.87±0.00	**0.58±0.01**
CNN-10(120)	AE_g	**3.27±0.04**	3.44±0.04	4.00±0.04	3.35±0.05
	AE_h	0.72±0.02	0.64±0.03	0.84±0.01	**0.55±0.01**
CNN-5-w(80)	AE_g	**3.84±0.01**	3.85±0.01	4.45±0.04	3.95±0.02
	AE_h	0.69±0.01	0.59±0.00	0.85±0.00	**0.54±0.01**
CNN-5-w(140)	AE_g	**2.98±0.01**	3.08±0.02	4.17±0.04	3.11±0.03
	AE_h	0.67±0.00	0.56±0.00	0.82±0.01	**0.50±0.00**
CNN-10-w(80)	AE_g	**3.70±0.01**	3.74±0.01	4.63±0.03	3.76±0.01
	AE_h	0.65±0.01	0.56±0.01	0.87±0.01	**0.48±0.01**
CNN-10-w(140)	AE_g	**2.90±0.01**	3.14±0.02	4.50±0.05	3.00±0.02
	AE_h	0.62±0.00	0.54±0.00	0.84±0.00	**0.47±0.00**
GRU(80)	AE_g	**4.08±0.01**	4.12±0.01	5.81±0.10	4.34±0.02
	AE_h	0.79±0.00	0.73±0.00	0.85±0.00	**0.61±0.00**
GRU(140)	AE_g	**3.49±0.02**	3.55±0.01	6.99±0.05	4.54±0.04
	AE_h	0.91±0.01	0.80±0.00	0.84±0.01	**0.64±0.00**

Table 1: Results of the experiments on DNA storage data. Contains all combinations of models, dimensions and losses. The results are reported in the format "$\mathrm{mean} \pm \mathrm{std}$" of the mean value and the standard deviation over 5 runs of the experiments.

80 and the ESDs corresponding to the engaged embedding networks. The results of these experimental settings are presented in Table 1. It is worth noting that, unlike in (Guo, Liang, and Hou 2022), we utilize a learnable scaling factor instead of a predefined fixed one. As a result, the obtained results may differ from those reported in (Guo, Liang, and Hou 2022), even though the overall setting appears similar.

As shown in Table 1, the proposed Poisson regression and PNLL achieve the best performance in terms of the metric AE_h across all the embedding networks. Although the performance of PNLL in AE_g is slightly behind the best performance, the relative difference is not significant. Moreover, as mentioned earlier, the AE_h is a more crucial metric of interest than AE_g in the experiments. These results indicate that the proposed Poisson regression outperforms other objective functions.

It can be also observed that increasing the embedding dimension from 80 to the ESD of the respective embedding network leads to better performance in both AE_h and AE_g, especially a significant improvement in AE_g. This finding highlights the effectiveness of the proposed ESD. However, for the GRU model, increasing the embedding dimension results in worse performance. It is speculated that the assumption A4 may not hold for the GRU model, as discussed in the Appendices. Overall, it is evident from Table 1 that the combination of the proposed PNLL and the proposed ESD yields the best performance.

Comparing models with different numbers of layers in the convolutional network, the improvement resulting from increasing the number of layers is found to be negligible. However, increasing the number of channels in the model leads to improved performance across most of the convolutional embedding networks. Furthermore, increasing the depth of the model results in a higher ESD, indicating that the model has a greater capacity to capture and express features from its input. Based on these observations, we can infer that width (number of channels) plays a more crucial role than depth (number of layers) in the task of Levenshtein distance embedding.

Readers may be concerned about computational overhead. Given that changes in embedding dimensions only affect the neural network's embedding top, the experimental time complexities are similar across different choices of embedding dimensions. Calculating the loss functions, as outlined in Equation (14) and Equation (17), is not computationally intensive. Therefore, the differences in time complexities due to the engaged loss functions are negligible.

In summary, the CNN-10-w embedding network, with an ESD embedding dimension of $n_0 = 140$ and optimized by the Poisson regression reports the highest performance among all the models.

Conclusion

Deep embedding is an effective way for fast approximation of Levenshtein distance. We propose a deep Levenshtein distance embedding algorithm based on Poisson regression. The implementation of our algorithm is based on two key points: embedding vector dimension and model training method. The adaptation of Poisson distribution to Levenshtein distance was theoretically analyzed under reasonable assumptions. Our method achieves a more accurate approximation on DNA storage data. Moreover, the proposed algorithm is robust to different embedding models and all show positive results. We hope our method will be useful with other approximate tasks that involve discrete labels.

Acknowledgements

The authors are grateful to the reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China under Grant 2020YFA0712100 and the National Natural Science Foundation of China.

References

Backurs, A.; and Indyk, P. 2015. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 51–58.

Bar-Lev, D.; Etzion, T.; and Yaakobi, E. 2023. On the Size of Balls and Anticodes of Small Diameter Under the Fixed-Length Levenshtein Metric. *IEEE Transactions on Information Theory*, 69(4): 2324–2340.

Berger, B.; Waterman, M. S.; and Yu, Y. W. 2020. Levenshtein distance, sequence comparison and biological database search. *IEEE transactions on information theory*, 67(6): 3287–3294.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, 737–744. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Chakraborty, D.; Goldenberg, E.; and Kouckỳ, M. 2016. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 712–725.

Chami, I.; Gu, A.; Chatziafratis, V.; and Ré, C. 2020. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in Neural Information Processing Systems*, 33: 15065–15076.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Church, G. M.; Gao, Y.; and Kosuri, S. 2012. Next-generation digital information storage in DNA. *Science*, 337(6102): 1628–1628.

Corso, G.; Ying, Z.; Pándy, M.; Veličković, P.; Leskovec, J.; and Liò, P. 2021. Neural distance embeddings for biological sequences. *Advances in Neural Information Processing Systems*, 34: 18539–18551.

Dai, X.; Yan, X.; Zhou, K.; Wang, Y.; Yang, H.; and Cheng, J. 2020. Convolutional embedding for edit distance. In *proceedings of the 43rd international ACM SIGIR conference on Research and Development in information retrieval*, 599–608.

Dong, Y.; Sun, F.; Ping, Z.; Ouyang, Q.; and Qian, L. 2020. DNA storage: research landscape and future prospects. *National Science Review*, 7(6): 1092–1107.

Erlich, Y.; and Zielinski, D. 2017. DNA Fountain enables a robust and efficient storage architecture. *science*, 355(6328): 950–954.

Goldman, N.; Bertone, P.; Chen, S.; Dessimoz, C.; LeProust, E. M.; Sipos, B.; and Birney, E. 2013. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *nature*, 494(7435): 77–80.

Grass, R. N.; Heckel, R.; Puddu, M.; Paunescu, D.; and Stark, W. J. 2015. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angewandte Chemie International Edition*, 54(8): 2552–2555.

Guo, A. J.; Liang, C.; and Hou, Q.-H. 2022. Deep Squared Euclidean Approximation to the Levenshtein Distance for DNA Storage. In *International Conference on Machine Learning*, 8095–8108. PMLR.

Haldar, R.; and Mukhopadhyay, D. 2011. Levenshtein distance technique in dictionary lookup methods: An improved approach. *arXiv preprint arXiv:1101.1232*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

Jiang, Y.; Li, G.; Feng, J.; and Li, W.-S. 2014. String similarity joins: An experimental evaluation. *Proceedings of the VLDB Endowment*, 7(8): 625–636.

Levenshtein, V. I.; et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, 707–710. Soviet Union.

Li, H.; and Homer, N. 2010. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5): 473–483.

Logan, R.; Fleischmann, Z.; Annis, S.; Wehe, A. W.; Tilly, J. L.; Woods, D. C.; and Khrapko, K. 2022. 3GOLD: optimized Levenshtein distance for clustering third-generation sequencing data. *BMC bioinformatics*, 23(1): 1–18.

Ostrovsky, R.; and Rabani, Y. 2007. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 54(5): 23–es.

Press, W. H.; Hawkins, J. A.; Jones Jr, S. K.; Schaub, J. M.; and Finkelstein, I. J. 2020. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Proceedings of the National Academy of Sciences*, 117(31): 18489–18496.

Qu, G.; Yan, Z.; and Wu, H. 2022. Clover: tree structure-based efficient DNA clustering for DNA-based data storage. *Briefings in Bioinformatics*, 23(5). Bbac336.

Rashtchian, C.; Makarychev, K.; Racz, M.; Ang, S.; Jevdjic, D.; Yekhanin, S.; Ceze, L.; and Strauss, K. 2017. Clustering billions of reads for DNA data storage. *Advances in Neural Information Processing Systems*, 30.

Sberro, H.; Fremin, B. J.; Zlitni, S.; Edfors, F.; Greenfield, N.; Snyder, M. P.; Pavlopoulos, G. A.; Kyrpides, N. C.; and Bhatt, A. S. 2019. Large-scale analyses of human microbiomes reveal thousands of small, novel genes. *Cell*, 178(5): 1245–1259.

Su, Z.; Ahn, B.-R.; Eom, K.-Y.; Kang, M.-K.; Kim, J.-P.; and Kim, M.-K. 2008. Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm. In *2008 3rd International Conference on Innovative Computing Information and Control*, 569–569. IEEE.

Wagner, R. A.; and Fischer, M. J. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1): 168–173.

Welzel, M.; Schwarz, P. M.; Löchel, H. F.; Kabdullayeva, T.; Clemens, S.; Becker, A.; Freisleben, B.; and Heider, D. 2023. DNA-Aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage. *Nature Communications*, 14(1): 628.

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. *Bioinformatics*, 35(11): 1820–1828.

Zorita, E.; Cuscó, P.; and Filion, G. J. 2015. Starcode: sequence clustering based on all-pairs search. *Bioinformatics*, 31(12): 1913–1919.

Details on the Embedding Networks

Five models, namely CNN-5, CNN-10, CNN-5-w, CNN-10-w, and GRU are utilized as the embedding networks in this study, following the architecture settings described in (Guo, Liang, and Hou 2022; Dai et al. 2020). The CNN-5 consists of five 1D-convolutional layers, while the CNN-10 has ten 1D-convolutional layers. Each 1D-convolutional layer has output channels of 64, a kernel size of 3, a stride of 1, and a padding size of 1. Average pooling layers with a kernel size of 2 and ReLU activation function are also applied. To generate the embedding vector, two fully connected layers and a final batch normalization layer are employed on top of the convolutional layers. The CNN-5-w and CNN-10-w are the wide versions of CNN-5 and CNN-10, respectively. The main difference between the wide versions and their original counterparts is that the number of channels in the convolutional layers is four times larger (256 vs. 64). The GRU model consists of two bidirectional recurrent layers with a hidden size of 64. Similar to the convolutional models, the GRU model also includes two fully connected layers and a final batch normalization layer at the top of the model.

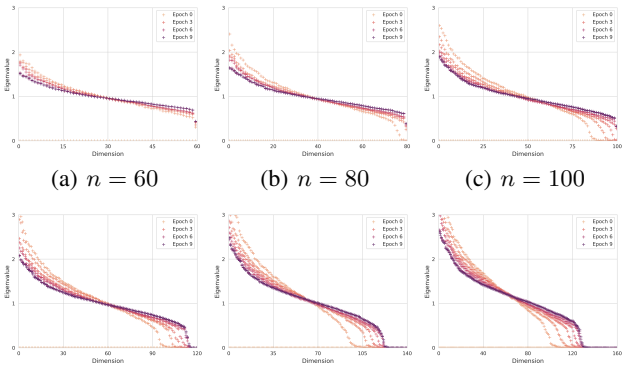
It is important to note that in the original work (Guo, Liang, and Hou 2022), the final batch normalization layer used a default parameter of $\epsilon = 1\mathrm{e}-5$ in PyTorch. This setting did not affect the normality of the elements of the embedding vectors in their study. However, by using the learnable scaling factor and increasing the embedding dimension, the normality is compromised. We discovered that the value of $\epsilon = 1\mathrm{e}-5$ is too large, and the model learns features with the same order of magnitude as ϵ. Consequently, the guarantee of the embedding elements following $N(0,1)$ was no longer valid. To address this issue, a smaller value of $\epsilon = 1\mathrm{e}-9$ for the final batch normalization layers is used in this research. This adjustment can be easily verified in the experiments. Since it is not our main focus, the details are not presented.

The Early Stopping Dimension for the Engaged Embedding Networks

In this appendix, we provide the sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i}-\boldsymbol{u_j}, \boldsymbol{u_i}-\boldsymbol{u_j}), (i \neq j)$ for different embedding dimensions. We also plot the approximation errors with respect to the embedding dimensions. These figures allow us to estimate the ESD values for the embedding networks CNN-10, CNN-5-w, CNN-10-w, and GRU.

CNN-10. For the embedding network CNN-10, the corresponding figures, namely Figures 3 and 4, exhibit similar patterns to those observed for CNN-5 in Figures 1 and 2. The estimated ESD is about $n_0 = 120$. The approximation error decreases as the embedding dimension increases until reaching the ESD n_0. The only difference between CNN-10 and CNN-5 is that the embedding network of CNN-10 demonstrates a lower standard deviation across different runs. Hence, we may infer that the embedding network of CNN-10 is more stable during the training phase compared to CNN-5.

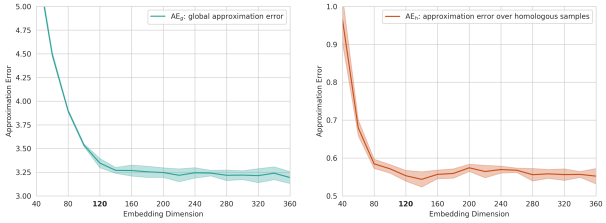
CNN-5**-w**. The results obtained from CNN-5-w are depicted in Figures 5 and 6. In comparison to Figure 1, it is



(a) $n = 60$ (b) $n = 80$ (c) $n = 100$

(d) $n = 120$ (e) $n = 140$ (f) $n = 160$

Figure 3: **CNN-**10. The sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i}-\boldsymbol{u_j}, \boldsymbol{u_i}-\boldsymbol{u_j}), (i \neq j)$ are plotted for different choices of the embedding dimension n for the CNN-10 embedding network. When n is small, the eigenvalues are distributed around 1, as in (a)–(c). Increasing the embedding dimension n, the sorted eigenvalues decrease to 0 after some dimension, as in (d)–(f).



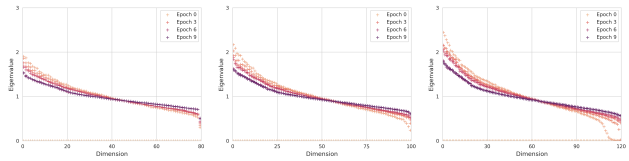
(a) AE_g (b) AE_h

Figure 4: **CNN-**10. The AE_g and AE_h are shown against the embedding dimension in (a) and (b), respectively. The curves are plotted based on the mean and standard deviation over 5 runs. The approximation errors decrease alongwith increase the embedding dimension n until the ESD n_0, which is 120 for CNN-10. When $n > n_0$, there is no gain of the performance on a larger n.

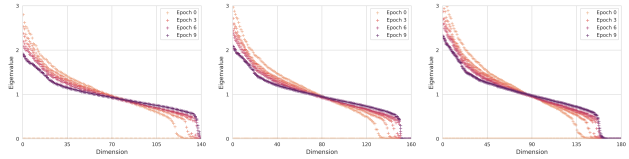
observed that the zero eigenvalues of $\mathrm{cov}(\boldsymbol{u_i}-\boldsymbol{u_j}, \boldsymbol{u_i}-\boldsymbol{u_j}), (i \neq j)$ starts to appear when the embedding dimension reaches 140 rather than 120. Based on this, the estimated ESD is about $n_0 = 140$. In Figure 6, the AE_g continues to decrease slightly after n_0, while the AE_h changes negligibly after n_0.

CNN-10**-w**. As suggested in Figures 7 and 8, the estimated ESD for the CNN-10-w engaged in this paper is approximately $n_0 = 140$. Similar to the difference between CNN-5 and CNN-5-w, the ESD n_0 of embedding network CNN-10-w exceeds that of the plain CNN-10.

GRU. As shown in Figure 9, the estimated ESD for the GRU embedding network is approximately $n_0 = 140$. However, unlike the sharp decline observed in the sorted eigenvalues of the convolutional networks, the sorted eigenvalues produced by the GRU embedding network exhibit a gradual decrease towards 0. This observation suggests that the

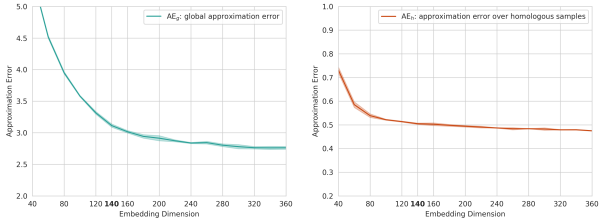


(a) $n = 80$ (b) $n = 100$ (c) $n = 120$



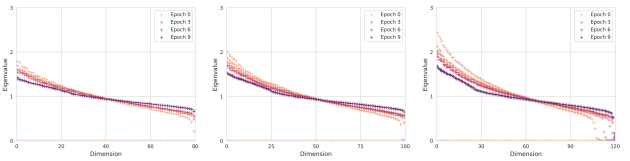
(d) $n = 140$ (e) $n = 160$ (f) $n = 180$

Figure 5: **CNN-5-w**. The sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$ are plotted for different choices of the embedding dimension n for the CNN-5-w embedding network. When n is small, the eigenvalues are distributed around 1, as in (a)–(c). Increasing the embedding dimension n, the sorted eigenvalues decrease to 0 after some dimension, as in (d)–(f).

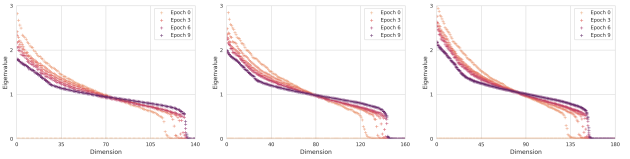


(a) AE_g (b) AE_h

Figure 6: **CNN-5-w**. The AE_g and AE_h are shown against the embedding dimension in (a) and (b), respectively. The curves are plotted based on the mean and standard deviation over 5 runs. The approximation errors decrease alongwith increase the embedding dimension n until the ESD n_0, which is 140 for CNN-5-w. When $n > n_0$, the improvement in performance with a larger n is not significant, particularly for AE_h.

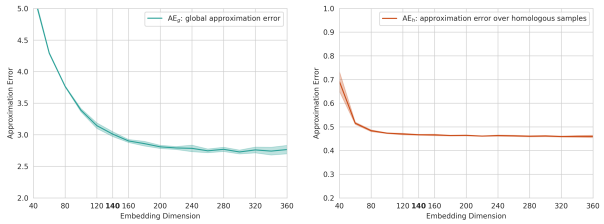


(a) $n = 80$ (b) $n = 100$ (c) $n = 120$



(d) $n = 140$ (e) $n = 160$ (f) $n = 180$

Figure 7: **CNN-10-w**. The sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$ are plotted for different choices of the embedding dimension n for the CNN-10-w embedding network. When n is small, the eigenvalues are distributed around 1, as in (a)–(c). Increasing the embedding dimension n, the sorted eigenvalues decrease to 0 after some dimension, as in (d)–(f).



(a) AE_g (b) AE_h

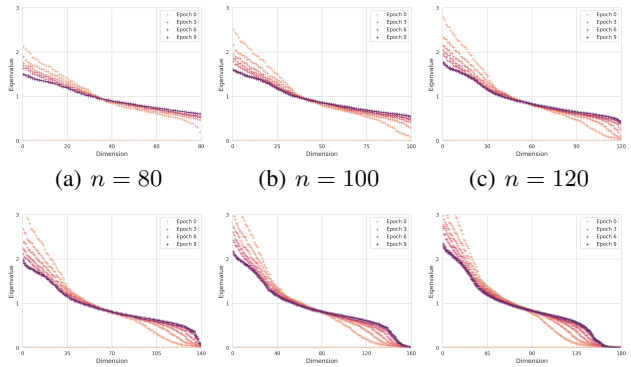
Figure 8: **CNN-10-w**. The global approximation error and homologous approximation error are shown against the embedding dimension in (a) and (b), respectively. The curves are plotted based on the mean and standard deviation over 5 runs. The approximation errors decrease alongwith increase the embedding dimension n until the ESD n_0, which is 140 for CNN-10-w. When $n > n_0$, the improvement in performance with a larger n is not significant, particularly for AE_h.

assumption stated in A4 may not hold for the GRU embedding network. Furthermore, Figure 10 demonstrates that the global approximation error reaches its minimum point before reaching the ESD $n_0 = 140$, while the homologous approximation error remains relatively stable with respect to the embedding dimension. It is also shown that the GRU does not provide a competitive performance, as in Figure 10 and Table 1. This suggests that the GRU may not be an optimal choice of embedding network.

When comparing the results of CNN-5, CNN-10, CNN-5-w, and CNN-10-w, as shown in Figures 1, 3, 5 and 7, it is observed that increasing the number of layers in the embedding network has little impact on enlarging the ESD, while enlarging the convolutional channels lead to a larger ESD. This indicates that a wider embedding network has a greater capacity to capture and express features from its input in the Levenshtein distance embedding job.

A Glimpse of Failed Sequences

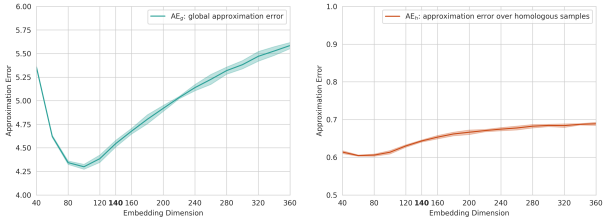
Certain kinds of sequences with low cardinality may pose challenges to the model, either due to their inherent property or imbalanced training samples. To investigate if there are sequences that consistently predict biased distances, we calculated the averaged predicted distance for each sequence $\boldsymbol{s}$ with a variable sequence $\boldsymbol{t}$ as $\mathrm{mean}_d(\boldsymbol{s}) = \mathrm{mean}\{\hat{d}(\boldsymbol{s}, \boldsymbol{t}) | \, d(\boldsymbol{s}, \boldsymbol{t}) = d\}$, and plotted the distribution of $\mathrm{mean}_d(\boldsymbol{s})$ over $\boldsymbol{s}$ for $d = 1, 2, 3$, as shown in Figure 11(a). In this figure, we found no evidence of such outlier sequences. We also printed and examined sequence pairs $(\boldsymbol{s}, \boldsymbol{t})$ leading to failed approximations. However, due to the length of the DNA sequences (around 150 nt) engaged in the ex-



(a) $n = 80$ (b) $n = 100$ (c) $n = 120$

(d) $n = 140$ (e) $n = 160$ (f) $n = 180$

Figure 9: **GRU**. The sorted eigenvalues of $\mathrm{cov}(\boldsymbol{u_i} - \boldsymbol{u_j}, \boldsymbol{u_i} - \boldsymbol{u_j}), (i \neq j)$ are plotted for different choices of the embedding dimension n for the GRU embedding network. When n is small, the eigenvalues are distributed around 1, as in (a)–(c). Increasing the embedding dimension n, the sorted eigenvalues decrease to 0 after some dimension, as in (d)–(f).



(a) AE_g (b) AE_h

Figure 10: **GRU**. The AE_g and AE_h are shown against the embedding dimension in (a) and (b), respectively. The curves are plotted based on the mean and standard deviation over 5 runs. The global approximation error reaches its minimal around embedding dimension $n = 100$ which is less than the ESD $n_0 = 140$. Meanwhile the homologous approximation error is relative stable with the embedding dimension.

periments, identifying clear patterns is challenging. To address this, we conducted experiments using shorter random sequences. The corresponding distribution of $\mathrm{mean}_d(\boldsymbol{s})$ on a random dataset of sequence length 7 is also shown in Figure 11(b), which similarly indicates no evidence of outlier sequences. Focusing on sequence pairs that give outlier approximations with Levenshtein distance 1, we observed a pattern, specifically, when insertion or deletion occurs at the beginning of a sequence with an extended homopolymer run (repeated letters), the predicted distance tends to be less accurate, as demonstrated in Table 2. Intuitively, we speculate that advanced network architectures, positional encoding, and bidirectional padding may help alleviate this issue.

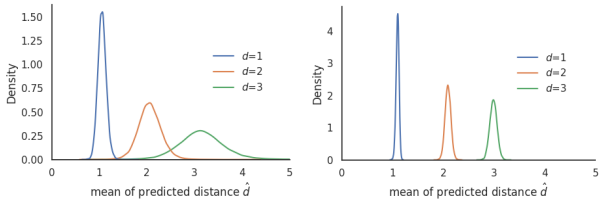


Figure 11: Distribution of $\mathrm{mean}_d(\boldsymbol{s})$. (a) the distribution is plotted using the DNA-storage dataset; (b) the distribution is plotted using a random dataset of sequences with length 7.

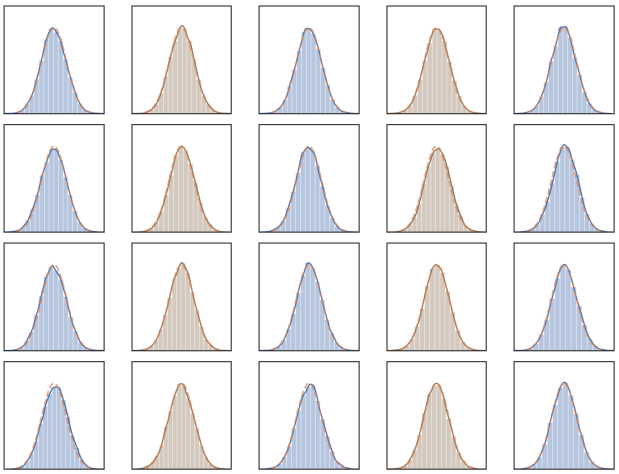
seq. pair	$\hat{d}$	seq. pair	$\hat{d}$
2 0 *3 3 3 3* 1 4 0 *3 3 3 3* 1 4 4	1.54	**2** *3 3 3 3 3 3* 4 *3 3 3 3 3 3* 4 4	1.52
1 1 **3** 1 *3 3 3* 4 1 1 1 *3 3 3* 4 4	1.51	*0 0 0* 3 0 3 3 4 **0** *0 0 0* 3 3 4 4	1.60
2 1 *3 3 3 3 3* 4 1 *3 3 3 3 3* 4 4	1.54	0 *2 2 2 2 2 2* 4 **1** 0 *2 2 2 2 2 2*	1.55
0 2 *0 0 0 0* 2 4 2 *0 0 0 0* 2 4 4	1.52	**2** 1 *3 3 3 3* 2 4 1 *3 3 3 3* 2 4 4	1.55
2 0 *3 3 3 3* 0 4 0 *3 3 3 3* 0 4 4	1.57	**2** 0 *2 2 2 2 2* 4 0 *2 2 2 2 2* 4 4	1.52
0 1 *0 0 0 0* 1 4 1 *0 0 0 0* 1 4 4	1.56	1 *3 3 3 3* 0 0 4 **2** 1 *3 3 3 3* 0 0	1.52
2 3 *1 1 1 1* 1 4 3 *1 1 1 1* 1 4 4	1.54		

Table 2: Outlier pairs with Levenshtein distance 1 from a random dataset of sequences with length 7. The boldface letters indicate the inserted or deleted letters; the italic letters show the extended homopolymer runs; the letter '4' is for the padding letter.

Assumption Verification

Primarily concerned assumptions are the assumptions: A1, A2, and A4. Among these, the A4 is a revision of A2. In Figures 1, 3, 5 and 7, the validation of A4 is confirmed. When the embedding dimension is below the early stop dimension n_0, the covariance matrices have eigenvalues close to 1, which indicates the embedding elements tend to be independent; when the embedding dimension exceeds n_0, the sorted eigenvalues exhibit a sharp decline to 0. This phenomenon verifies the A4 to some extent.

To show whether A1 holds when the embedding dimension varies, we plotted the statistical distributions of the first several embedding elements, as illustrated in Figure 12. This figure suggested that the embedding dimension has minimal impact on the normality of embedding elements $\boldsymbol{u}_i$.



(a) $n = 40$ (b) $n = 80$ (c) $n = 120$ (d) $n = 160$ (e) $n = 200$

Figure 12: The distribution of the first 4 embedding elements by CNN-10-w. (a)-(e) correspond to different choices of embedding dimension from $n = 40$ to $n = 200$.