Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Original Paper

OXFORD

Sequence Analysis

SENSE: Siamese neural network for sequence embedding and alignment-free comparison

Wei Zheng[1,†], Le Yang[1,†], Robert J. Genco[2, 3], Jean Wactawski-Wende[4], Michael Buck[5], Yijun Sun[1, 3*]

[1]Department of Computer Science and Engineering, [2]Department of Oral Biology, [3]Department of Microbiology and Immunology, [4]Department of Epidemiology and Environmental Health, [5]Department of Biochemistry, University at Buffalo, The State University of New York, Buffalo, NY 14214

*To whom correspondence should be addressed. Zheng and Yang contributed equally to this work.

Associate Editor: Alfonso Valencia

Received on April 16, 2018; revised on September 3, 2018; accepted on October 14, 2018

Abstract

Motivation: Sequence analysis is arguably a foundation of modern biology. Classic approaches to sequence analysis are based on sequence alignment, which is limited when dealing with large-scale sequence data. A dozen of alignment-free approaches have been developed to provide computationally efficient alternatives to alignment-based approaches. However, existing methods define sequence similarity based on various heuristics and can only provide rough approximations to alignment distances.
Results: In this paper, we developed a new approach, referred to as SENSE (**S**iam**E**se **N**eural network for **S**equence **E**mbedding), for efficient and accurate alignment-free sequence comparison. The basic idea is to use a deep neural network to learn an explicit embedding function based on a small training dataset to project sequences into an embedding space so that the mean square error between alignment distances and pairwise distances defined in the embedding space is minimized. To the best of our knowledge, this is the first attempt to use deep learning for alignment-free sequence analysis. A large-scale experiment was performed that demonstrated that our method significantly outperformed the state-of-the-art alignment-free methods in terms of both efficiency and accuracy.
Availability and implementation: Open-source software for the proposed method is developed and freely available at https://www.acsu.buffalo.edu/~yijunsun/lab/SENSE.html.
Contact: `yijunsun@buffalo.edu`
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Sequence analysis is a major research area in bioinformatics and has a wide range of applications in database search, sequence annotation, metagenomics, comparative genomics, and gene prediction. Classic approaches to sequence analysis are based on sequence alignment, either global or local, pairwise or multiple sequence alignment. In this paper, we focus on global pairwise sequence alignment, for which the Needleman-Wunsch (NW) algorithm is the optimal method (Needleman and Wunsch, 1970). A major limitation of the NW algorithm is its high computational complexity and thus is very limited when dealing with large-scale sequence data. With the advent of next-generation sequencing technologies, the data generation capacity has increased dramatically at a speed exceeding Moore's law and with sharply reduced cost. The rapid accumulation of sequence data poses a serious challenge for data analysis, demanding new computational algorithms for efficient data processing.

Over the past two decades, a dozen of alignment-free approaches have been developed to provide computationally efficient alternatives to alignment-based approaches, and found a wide range of applications in database search, sequence annotation, metagenomics, comparative genomics, and gene prediction (see Zielezinski *et al.* (2017); Bonham-Carter *et al.* (2013); Song *et al.* (2013) for in-depth reviews). Commonly used methods can be broadly classified into two categories: 1) methods based on word frequency (e.g., k-mer (Karlin and Burge, 1995), FFP (Sims *et al.*, 2009), CV (Gao and Qi, 2007)), and 2) methods based on sub-strings (e.g., ACS (Ulitsky *et al.*, 2006), Kr (Haubold *et al.*, 2009), kmacs

(Leimeister and Morgenstern, 2014)). There also exist some methods based on information theory (e.g., IC-PIC (Gao and Luo, 2012)), but they are less commonly used. The methods in the first category are based on the statistics of fixed-length word frequency or on the information content of word frequency distributions, while the methods in the second category employ the similarities or differences of sub-strings in a pair of sequences. For sub-string methods, it is not necessary to specify a word length, and thus in general they can achieve better performance than those relying on a fixed word length. However, sub-string methods introduce new parameters (e.g., the number of mismatches in kmacs (Leimeister and Morgenstern, 2014)) that cannot be easily estimated. Moreover, computing features of variable word lengths usually requires more complex data structures and thus is computationally much more expensive (Leimeister and Morgenstern, 2014). In addition to the aforementioned issues, another major limitation of existing methods is that they are all data-independent approaches, where distance measures are defined based on various heuristics and thus can only provide rough approximations to alignment distances.

In this paper, we propose a new method, referred to as SENSE (SiamEse Neural network for Sequence Embedding), for efficient and accurate alignment-free sequence comparison. The basic idea is to use a deep neural network to learn an explicit embedding function and map sequences onto an embedding space so that the mean square error between alignment distances and pairwise distances defined in the embedding space is minimized. We developed methods that allow researchers to select a small fraction of sequence data to train the constructed model and to estimate the dimension of an embedding space. To the best of our knowledge, this is the first attempt to use deep learning for alignment-free sequence comparison. Compared to the existing alignment-free methods, our method offers a number of advantages: (*i*) SENSE is a supervised learning method where the embedding function is learned *automatically* through training, while in the existing methods sequence similarities are defined based on heuristics. Consequently, our method is much more accurate than the existing methods. A large-scale experiment was performed on real-world datasets that demonstrated that the mean square errors of our method are one to two orders of magnitude smaller than other methods. (*ii*) Our method is computationally very efficient, and runs even faster than the vanilla k-mer method. (*iii*) Our method is largely insensitive to the specific choice of the parameters, making parameter tuning and hence the implementation of our method easy for users. Considering the wide applications of alignment-free methods, particularly k-mer, in sequence analysis, we believe that this work opens the door to develop a range of methods that do not rely on sequence alignment for large-scale sequence data analysis.

2 Methods

In this section, we present the proposed method for sequence embedding. For ease of implementation, we focus on the analysis of amplicon sequence data, which has roughly equal sequence lengths. We start by proving that the k-mer method can be viewed as a simple, untrainable neural network, which motivated the development of the proposed method.

2.1 k-mer Method

The k-mer method (Karlin and Burge, 1995) is probably the most commonly used alignment-free method for sequence comparison, and serves as the basis for a wide range of bioinformatics methods (e.g., ESPRIT (Sun *et al.*, 2009) and SLAD (Zheng *et al.*, 2018) for sequence binning, RDP classifier (Wang *et al.*, 2007) and Kraken (Wood and Salzberg, 2014) for sequence annotation, to name a few) and other alignment-free methods (e.g., FFP (Sims *et al.*, 2009), CV (Gao and Qi, 2007)). Given an alphabet $\Omega = \{\text{A, T, C, G}\}$ and a pre-defined number k, it proceeds by first constructing a dictionary consisting of all possible sub-sequences of length k, then sliding each sub-sequence (also called k-mer) against a sequence, and constructing a counting vector where each entry records the number of the corresponding k-mer detected in the sequence.

We show that the k-mer method can be implemented as an one-layer convolutional neural network (Supplementary Fig. 1). To see this, we first transform an input sequence of length L into an $L \times 4$ matrix by using the one-hot encoding, where each row represents a nucleotide A, T, C or G. Similarly, each k-mer can be encoded as a $k \times 4$ filter matrix. Then, we convolve each filter through the input sequence matrix starting from position 1 to position $L - k + 1$, and feed the output of a filter at each position into an activation function defined as $f(x) = \max(0, x-k+1)$, which yields a value of 1 or 0 indicating whether a k-mer pattern is detected. Since there are a total of 4^k filters, the convolution process generates 4^k binary vectors of length $L-k+1$. Finally, a flatten layer is generated that concatenates all the binary vectors into one vector, and then fully connected to an output layer consisting of 4^k nodes. However, the weights between the two layers are fixed, taking a value of 1 between the binary vector generated by the i-th filter and the i-th node and 0 otherwise. The network is not trainable, since the input and output are not directly comparable.

2.2 Siamese Neural Network for Sequence Embedding

Given an alphabet Ω and a set of sequences of length L defined over Ω, we aim to find an embedding function $\psi : \Omega^L \rightarrow \mathcal{R}^d$ that maps the sequences into a d-dimensional space so that the difference between the alignment distances and the pairwise distances defined in the embedding space are minimized. We propose to use neural network to learn the explicit embedding function, by leveraging the expressive power of neural network to approximate complex functions (Csáji, 2001). The k-mer method provides one way for sequence embedding. However, as shown above, it is actually a shallow, untrainable neural network with only one convolution layer. Recent developments in deep learning have demonstrated that models with multiple layers can significantly improve learning accuracy compared to shallow models (LeCun *et al.*, 2015). Training a deep neural network usually requires a large number of labeled samples. As we will see shortly, with the advent of next-generation sequencing technology and the design of our method, we have more than enough data for training a deep learning model. The above observations motivated us to develop the SENSE method presented below.

2.2.1 Siamese Neural Network

Fig. 1(a) presents an overview of the proposed method and its training process. Embedding data into a lower dimensional space or dimensionality reduction more generally has been intensively studied in machine learning. However, most work was performed on numerical data, where autoencoder is the most commonly used neural network (LeCun *et al.*, 2015). In our application, inputs and outputs are from the sequence and numerical domains, respectively, and thus are not directly comparable. To address this issue, we propose to use the Siamese neural network (Bromley *et al.*, 1994) to learn an explicit embedding function.

Siamese neural network is a class of network architectures that consist of two identical networks, taking a sequence pair as one training sample. Given a pair of sequences (s_i, s_j), each network takes one sequence as input and outputs $\psi(s_i|\mathcal{W})$ and $\psi(s_j|\mathcal{W})$ as the embedding vectors of the two sequences, respectively. Here, $\mathcal{W}$ is the parameters of the network to be optimized. Then, we compute the alignment distance $d_a(s_i, s_j)$ by using the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), and the embedding distance $d_e(\psi(s_i), \psi(s_j))$. For the reasons that will be clear shortly, the embedding distance is calculated as the generalized Jaccard distance (Levandowsky and Winter, 1971). Ideally, the difference between $d_a(s_i, s_j)$ and $d_e(\psi(s_i), \psi(s_j))$ should be as small as possible.

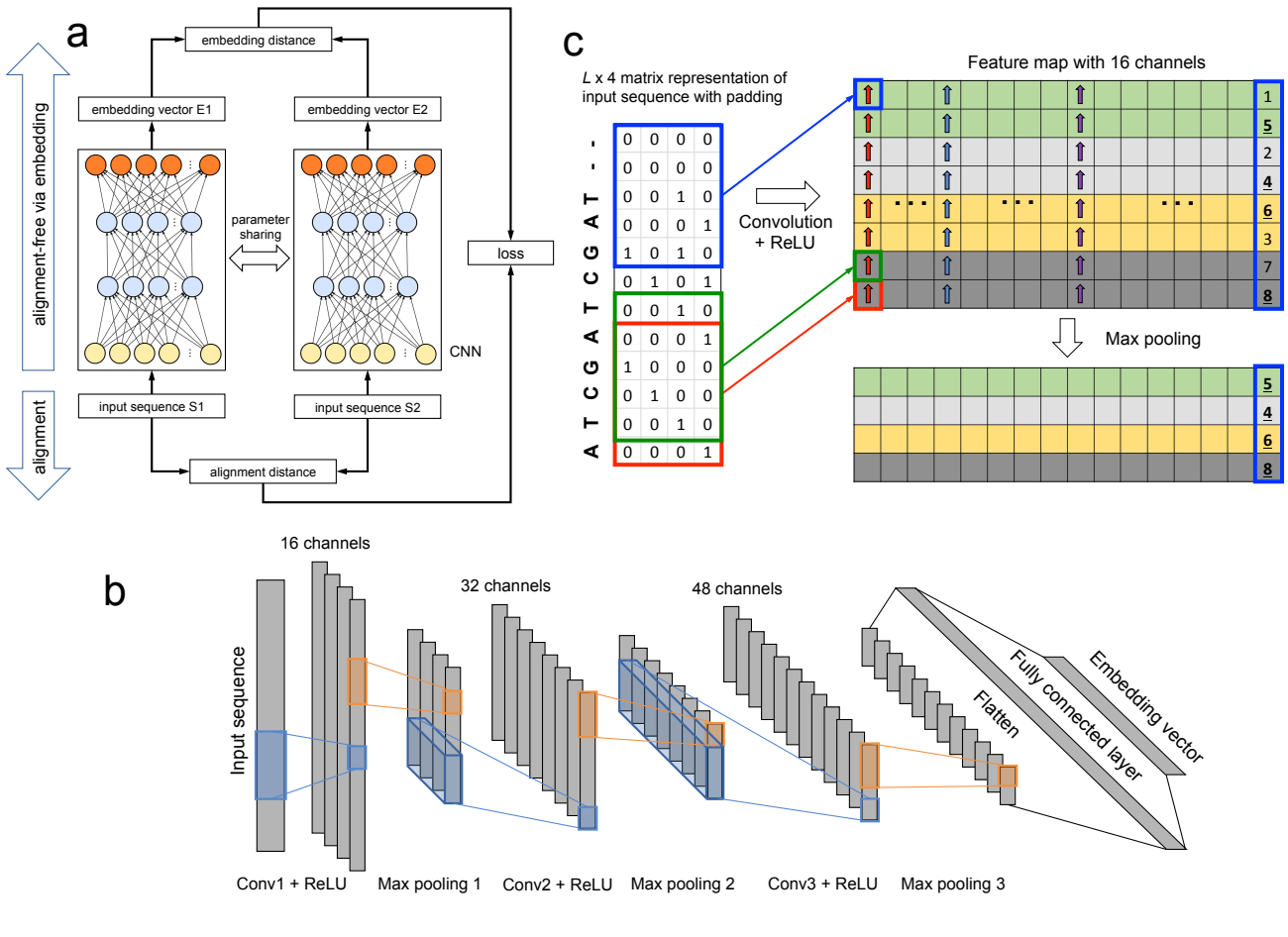


Fig. 1. Overview of the proposed SENSE method for sequence embedding and its training process. (a) Siamese neural network consists of two identical networks that take two sequences as input and output two embedding vectors. The network parameters are shared by the two networks and learned by minimizing the mean square error measuring the difference between alignment distances and embedding distances. (b) Detailed structure of a three-layer convolutional neural network (CNN) that converts a biological sequence into a numeric vector. (c) Toy example illustrating how a convolution operation and max pooling are performed on a sequence.

Hence, we train Siamese neural network using back-propagation (LeCun *et al.*, 1989) to minimize the mean square error given by

$$\mathcal{L}(\mathcal{W}) = \sum_{i,j} \left(d_a(s_i, s_j) - d_e\left(\psi(s_i|\mathcal{W}), \psi(s_j|\mathcal{W})\right)\right)^2 \ . \quad (1)$$

In the training process, the twin networks are forced to share the same parameters, including both initialization and gradient descent updates. Once the Siamese neural network is optimized, one of the networks can be used for sequence comparison.

2.2.2 Convolutional Neural Network

This section discusses the design of the neural network used in the twin networks. Currently, convolutional neural network (CNN) and recurrent neural network (RNN) are two most commonly used network architectures for deep learning (LeCun *et al.*, 2015). CNN is a feed-forward neural network and has achieved exceptional results in many applications, particularly in image recognition and text mining (LeCun *et al.*, 2015; Dos Santos and Gatti, 2014; Krizhevsky *et al.*, 2012). For our applications, it is required that input sequences have the same length. In contrast, by connecting nodes to form a directed cycle, RNN is able to detect dynamic spatial patterns and process sequences of various lengths (Hochreiter and Schmidhuber, 1997). However, training RNN is generally much more difficult than CNN. In this study, we focus mainly on amplicon sequence analysis, where sequences usually have fairly stable lengths and the problem of limited variations in read lengths can be readily solved by using zero-padding and trimming tricks. Moreover, considering that k-mer is a simple CNN network, we used CNN to form the twin networks, leaving it to future studies to use RNN to develop a model to process sequences of various lengths.

Fig. 1(b) depicts the detailed structure of our designed convolutional neural network. It consists of three convolution modules followed by a flatten layer and a fully connected layer. Each convolution module contains a convolution layer, a ReLU layer, and a max-pooling layer. Given an input sequence, we first transform it into a $L \times 4$ matrix by using the one-hot coding and then feed the matrix into a convolution layers (Fig. 1(c)). The convolution layer consists of a set of learnable filters, which have a small receptive field and are convolved through the full depth of each feature channel. For image data, 2-dimensional convolution and max-pooling are performed through both width and length directions, while for sequence data, the two operations scan through only the length direction. The output of a convolution layer is called a feature map. Usually, one convolution layer has multiple filters, each generating one channel of a feature map. Each entry of the feature map is then fed to the ReLU function, which is the most commonly used activation function (Nair and Hinton, 2010). The output of a ReLU layer is then passed to a max-pooling layer, which partitions each channel of a feature map into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. It provides a form of non-linear downsampling to reduce the number of parameters

Algorithm 1: Active-Landmark-Selection ($\mathcal{S},\ r,\ m$)

Input: data $\mathcal{S} = \{s_1, \cdots, s_N\}$, the number of landmarks r, the number of candidates m

Output: training data $\mathcal{T}$

randomly select s_{l_0} from $\mathcal{S}$;
$\mathcal{K} = \{s_{l_0}\}$;
$\mathcal{T} = \{\}$;
$\mathcal{D}_{min} = \{u_1 = 1, \cdots, u_m = 1\}$;
for $i = 0$ **to** $r - 1$ **do**
  randomly select $\mathcal{C} = \{s_{c_1}, \cdots, s_{c_m}\}$ from $\mathcal{S}$;
  for $j = 1$ **to** m **do**
    $u_j = \min(d_a(s_{c_j},\ s_{l_i}), u_j)$;
    $\mathcal{D}_{sort} = \text{sort}(\mathcal{D}_{min})$;
    randomly sample u_k from $\mathcal{D}_{sort}[\lfloor \frac{m}{2} \rfloor, \cdots, m]$;
    /* $\lfloor\cdot\rfloor$ is the floor function */
    $l_{i+1} = k$;
    $\mathcal{K} = \mathcal{K} \cup \{s_{l_{i+1}}\}$;
    $\mathcal{T} = \mathcal{T} \cup \{(s_{c_j},\ s_{l_i}),\ d_a(s_{c_j},\ s_{l_i})\}$;
  end
end

and amount of computation in the network, and hence to also control overfitting. After a sequence is processed by three convolution modules, it is forwarded to the flatten layer, where the feature map from the previous layer is concatenated into one vector. Finally, the flatten layer is fully connected to the output layer that generates an embedding vector.

It is now clear that our method can be viewed as a sophisticated extension of the traditional k-mer method. By using a multi-layer architecture, SENSE has the potential to build high-level features of input sequences that may not be attainable by a shallow model (Lee *et al.*, 2009). More importantly, unlike the k-mer method where the filters are handcrafted and the weights are fixed, the filters and weights in our method are all learned automatically through training. Thus, it is expected that SENSE performs much better than k-mer, which is demonstrated in our experiment.

2.2.3 Network Implementation

In our implementation, we set the number of nodes in the input layer as the maximum length of input sequences, the filter length to 5, and the numbers of filters in the three convolution layers to 16, 32 and 48, respectively. Consequently, the sizes of filters in the three convolution layers are 5×4, 5×16, and 5×32, respectively. For all convolution layers, we set padding = 2 and stride = 1 so that the output feature map has the same length as its input. For the max-pooling layers, we set window size = 2 and stride = 2. The number of nodes in the output layer (i.e., the dimension of embedding vectors) is set be equal to the length of input sequences. In the Experiments section, we provided a way to estimate the dimension of an embedding space. We implemented our method in Pytorch, which is a deep-learning framework that provides tensors and dynamic neural networks in Python.

2.2.4 Calculating Embedding Distances

Theoretically, any distance function can be used to measure the similarity between two embedding vectors. In our method, the loss is computed as the difference between alignment distances and embedding distances. The alignment distance is defined as the number of mismatches divided by the total alignment length, which is a number between 0 and 1. In order to make embedding distances comparable to alignment distances, the generalized Jaccard distance was used to measure the dissimilarity between two embedding vectors. Specifically, given vectors $\mathbf{x} \geq 0$ and $\mathbf{y} \geq 0$, the generalized Jaccard distance is defined as:

$$d_e(\mathbf{x}, \mathbf{y}) = 1 - \sum_i \min\left(x(i), y(i)\right) / \sum_i \max(x(i), y(i)) \ , \quad (2)$$

where $x(i)$ is the i-th element of $\mathbf{x}$. Note that the k-mer distance is essentially the generalized Jaccard distance between two k-mer counting vectors. The max and min operations can be easily implemented by using the Maxout network (Goodfellow *et al.*, 2013).

2.2.5 Active Landmark Selection

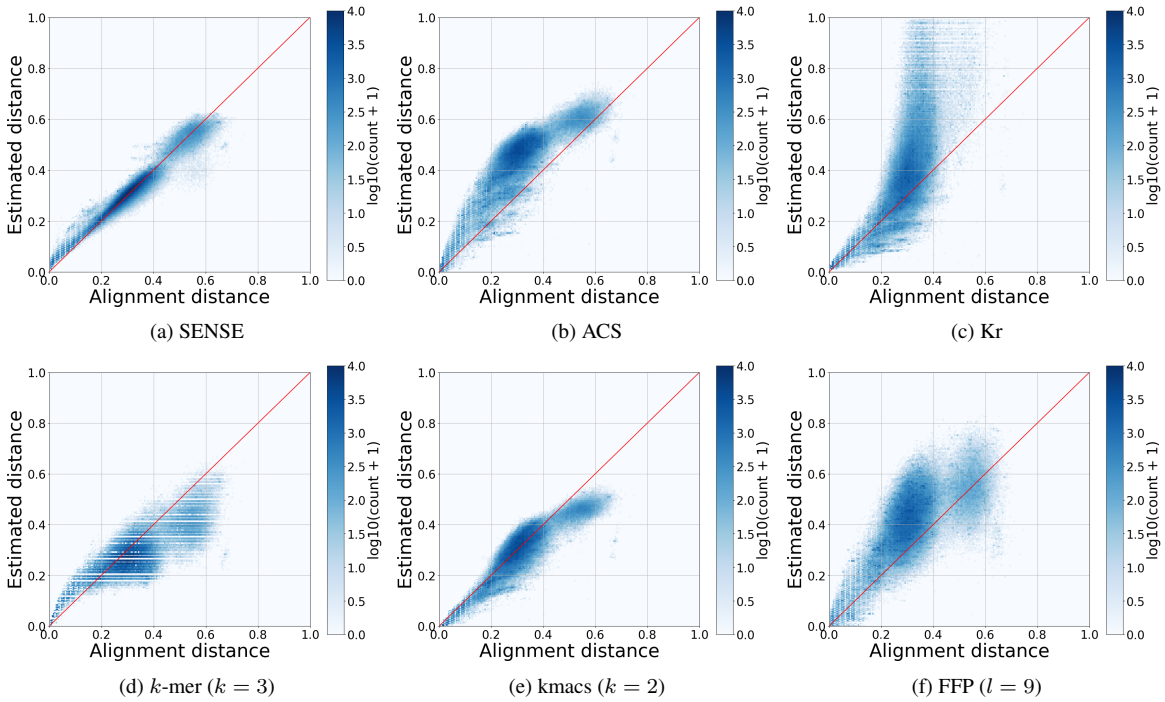
This section addresses the issue of how to select training samples for a large sequence dataset. While having an access to sufficient data for training is a motivation for us to using deep learning for sequence embedding, the number of available training samples can become excessively large. Given a sequence dataset with N sequences, the number of possible sequence pairs (i.e., training samples) is $N(N-1)/2$. If N is on the order of $10^6 \sim 10^9$, it is computationally unfeasible to train a model on such a large amount of data. One possible way to address the issue is to use a subset of sequences selected through random sampling, which however could bias the selection process toward dense regions. In order to select sequences that cover the entire data space, we propose to use an active landmark selection based method to select training samples. Similar techniques have been successfully used for large-scale clustering analysis (Voevodski *et al.*, 2012; Cai and Sun, 2011; Mao *et al.*, 2015). Algorithm 1 presents the pseudo-code of the sample selection method. Specifically, given a sequence dataset $\mathcal{S} = \{s_1, \cdots, s_N\}$, we start by randomly selecting a sequence s_{l_0} from the dataset to form a landmark set $\mathcal{K}$. Then, we randomly sample m candidate sequences and compute the distance between each sampled sequence and the landmark set $\mathcal{K}$. Here, the distance between a sequence and a landmark set is defined as the minimum distance between the sequence and a landmark sequence. We maintain a list $\mathcal{D}_{min}$ of length m to keep track of the minimum distances. Afterward, we randomly select a new landmark from the $m/2$ sequences that are the farthest from the landmark set and put it into landmark set $\mathcal{K}$. The selection procedure is repeated until r landmark sequences are selected. During the landmark selection process, all compared sequence pairs are collected as training data for the SENSE method. As we will see shortly in our experiment, given a dataset with 10^6 sequences, only a small fraction of sequences ($\sim$0.1%) are needed to train a model. In our software package, we implemented the training data selection algorithm with C++.

2.2.6 Sample Weighting

In the above derivations, we assume that all sequence pairs are equally important, which may not be the case in real applications. For example, for database search, researchers are interested only in the best matched reference sequences in a database for a query sequence. For microbiome research, most data analyses are performed at the species and genus levels (Sun *et al.*, 2009, 2010; Cai *et al.*, 2017), where sequences with 3% and 5% dissimilarities are assumed to be from the same species and genus, respectively. Unlike existing alignment-free methods, our method is a supervised learning based method. It provides researchers with a flexible way to force the method to learn a model that predicts more accurately on similar sequence pairs by assigning higher weights on those pairs. Specifically, given embedding distance d_e and alignment distance d_a, we can define a loss function as

$$\mathcal{L}(d_a, d_e) = w(d_a)(d_a - d_e)^2 \ , \quad (3)$$

in favor of distance estimation for similar sequence pairs, where w can be either a pre-defined constant or even a function of alignment distances.



(a) SENSE (b) ACS (c) Kr

(d) k-mer ($k = 3$) (e) kmacs ($k = 2$) (f) FFP ($l = 9$)

Fig. 2. Visualization of alignment distances versus estimated distances computed by six alignment-free methods performed on the Qiita dataset. The hexagon-bin plots were generated by using python code *matplotlib.pyplot.hexbin* and the number of bins in the x-direction was set to 200. The color of a bin represents the number of sequence pairs in the bin.

2.3 Related Work

We compared our method with k-mer and four state-of-the-art alignment-free methods, namely ACS (Ulitsky *et al.*, 2006), Kr (Haubold *et al.*, 2009), FFP (Sims *et al.*, 2009) and kmacs (Leimeister and Morgenstern, 2014). Below, we give a brief review of each method.

FFP is closely related to k-mer. It works by first calculating the count of each possible k-mer in a sequence and then dividing the k-mer counting vector by the total k-mer counts to convert a sequence into a feature frequency profile (FFP). The pairwise distance between two sequences is then defined as the Jensen–Shannon divergence of their respective FFPs. In ACS, given a pair of sequences s_a and s_b, the longest substring in s_a starting at positions i that exactly matches some substrings in s_b is identified and the length of the detected substring is recorded in $l_a(i)$. Then, a similarity measure between two sequences is computed as

$$L(s_a, s_b) = 1/|s_a| \sum_{i=1}^{|s_a|} l_a(i), \tag{4}$$

where $|s_a|$ is the length of s_a. Intuitively, the larger $L(s_a, s_b)$ is, the more similar the two sequences are. To account for the differences in sequence lengths, $L(s_a, s_b)$ is normalized as $L(s_a, s_b)/\log|s_a|$. To derive a distance measure, the inverse of similarity measure is taken and a correction term is subtracted to ensure that $d(s_a, s_a) = 0$, which yield

$$d(s_a, s_b) = \frac{\log|s_b|}{L(s_a, s_b)} - \frac{\log|s_a|}{L(s_a, s_a)}. \tag{5}$$

Since $d(s_a, s_b)$ is not symmetric, ACS computes $d_{ACS}(s_a, s_b) = (d(s_a, s_b) + d(s_a, s_b))/2$ and uses it as the final distance measure between two sequences. The kmacs method is a generalization of ACS. Briefly, to define the distance between two sequences s_a and s_b, kmacs searches for each position i in s_a the longest substring starting at i and matching some substring in s_b with up to k mismatches. It uses the average length of the longest substrings as a measure of similarity between two sequences and turns it into a symmetric distance measure in the same way as ACS. However, since searching for the longest substring with exact k mismatches is computationally very expensive, kmacs uses approximations instead of computing exact k-mismatch substrings. Kr is also closely related to ACS, which calculates the number of substitutions per site between two sequences using the shortest absent substring.

In a broad sense, our work is related to metric learning (Bellet *et al.*, 2013). Given a set of feature vectors, metric learning aims to learn a metric function by optimizing a certain cost function. For sequence and image analysis, feature vectors are usually handcrafted through so-called feature engineering, which is sometimes difficult and requires expert knowledge, and the metric function to be learned is commonly set to be Mahalanobis distance (Xing *et al.*, 2003). A major advantage of our method is that it can learn both features and an arbitrary embedding function (not necessarily a metric function) represented by a deep neural network from data simultaneously. Since metric learning-based methods are not as widely used as the above discussed alignment-free methods for sequence data analysis, we do not compare them with our method in the experiment.

3 Experiments

We performed a large-scale experiment that demonstrated that the proposed method significantly outperformed the state-of-the-art alignment-free methods in terms of both accuracy and efficiency.

3.1 Datasets

Two real-world sequence datasets were used in the experiment. The first dataset, referred to as Qiita, was downloaded from Qiita (study # 10052) (Clemente *et al.*, 2015). It contains 66 skin, saliva and feces samples

Table 1. CPU time (in second) and MSE results averaged over ten runs for six methods performed on the Qiita and RT988 datasets. The numbers in parentheses are standard deviations. When different parameters were used for a method, the best result is boldfaced. A *p*-value was computed by using Student's *t*-test to compare the MSE result of SENSE with the best result of a method. As a reference, the CPU time of the Needleman-Wunsch (NW) algorithm is also reported.

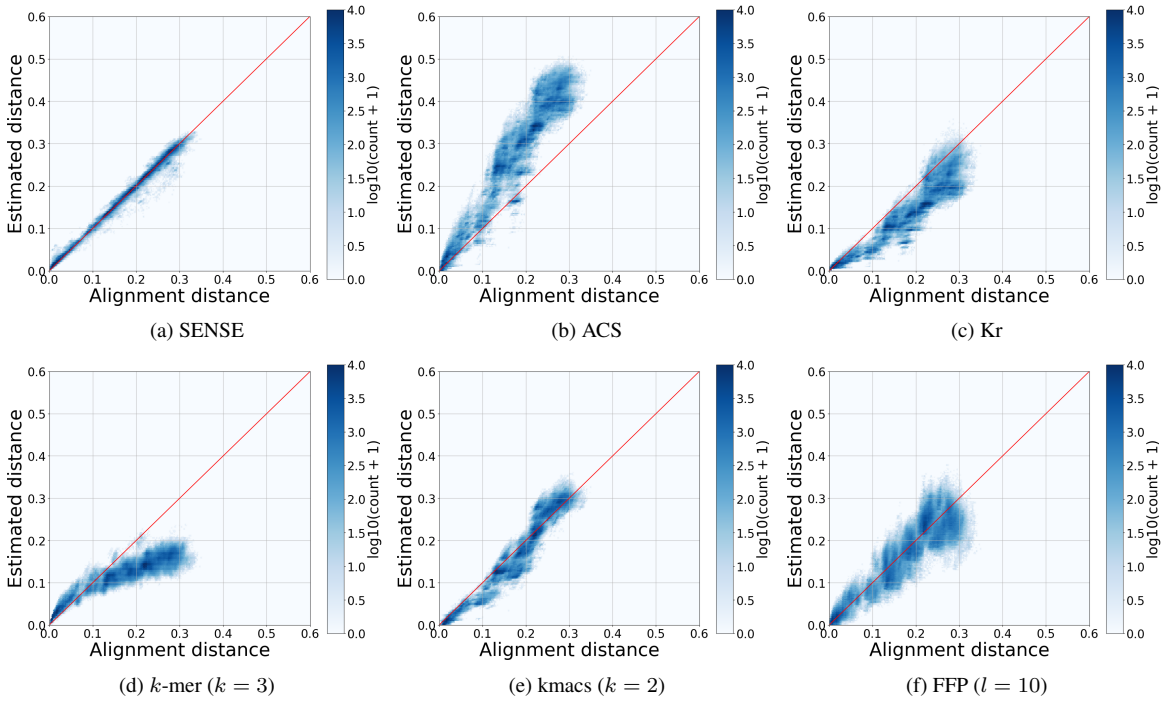
		Qiita			RT988		
Method	**Parameter**	**CPU time**	**MSE**	***p*-value**	**CPU time**	**MSE**	***p*-value**
SENSE	default	5.2 (0.1)	**3.4e-04 (1.1e-05)**	–	13.6 (0.2)	**2.1e-05 (1.3e-06)**	–
ACS	default	62.4 (1.2)	**2.2e-02 (1.3e-04)**	8.7e-39	182.3 (6.4)	**1.2e-02 (1.6e-04)**	3.3e-32
Kr	default	425.8 (7.6)	**1.1e00 (6.0e-02)**	9.1e-22	758.3 (14.6)	**2.8e-03 (4.7e-05)**	1.1e-30
k-mer	$k=3$	8.2 (0.2)	**4.5e-03 (1.7e-04)**	1.2e-23	10.6 (0.9)	4.9e-03 (1.4e-04)	
	$k=4$	13.7 (0.3)	3.2e-02 (3.0e-04)		23.7 (1.1)	**3.3e-03 (4.8e-05)**	8.2e-32
	$k=5$	16.5 (0.3)	1.4e-01 (5.2e-04)		40.2 (2.3)	3.8e-02 (4.5e-04)	
	$k=6$	17.0 (0.3)	2.3e-01 (5.1e-04)		49.0 (2.5)	9.9e-02 (1.2e-03)	
	$k=7$	17.1 (0.3)	3.0e-01 (6.4e-04)		51.7 (2.8)	1.5e-01 (1.7e-03)	
	$k=8$	16.9 (0.4)	3.4e-01 (7.7e-04)		52.5 (3.1)	1.9e-01 (2.0e-03)	
	$k=9$	16.7 (0.3)	3.4e-01 (7.7e-04)		52.7 (3.5)	1.9e-01 (2.0e-03)	
	$k=10$	16.7 (0.3)	3.4e-01 (7.6e-04)		52.9 (3.8)	1.9e-01 (2.0e-03)	
	$k=11$	16.6 (0.4)	3.4e-01 (7.6e-04)		53.3 (5.1)	1.9e-01 (2.0e-03)	
	$k=12$	16.7 (0.4)	3.4e-01 (7.6e-04)		52.7 (3.5)	1.9e-01 (2.0e-03)	
kmacs	$k=1$	93.9 (6.3)	4.5e-03 (3.6e-05)		279.9 (18.3)	1.9e-03 (3.9e-05)	
	$k=2$	102.4(6.2)	**1.7e-03 (2.8e-05)**	9.0e-29	315.9 (17.8)	**4.8e-04 (1.2e-05)**	1.4e-27
	$k=3$	110.1 (3.8)	3.8e-03 (6.0e-05)		344.7 (18.7)	1.3e-03 (1.5e-05)	
	$k=4$	118.1 (4.4)	7.6e-03 (8.4e-05)		371.8 (21.9)	2.9e-03 (2.5e-05)	
	$k=5$	127.2 (7.1)	1.2e-02 (1.0e-04)		394.1 (17.9)	4.6e-03 (3.1e-05)	
	$k=6$	133.2 (4.7)	1.6e-02 (1.2e-04)		415.6 (12.3)	6.1e-03 (4.2e-05)	
	$k=7$	138.6 (4.7)	1.9e-02 (1.3e-04)		432.2 (9.2)	7.7e-03 (6.5e-05)	
	$k=8$	144.4 (4.9)	2.3e-02 (1.4e-04)		450.1 (14.4)	9.1e-03 (8.7e-05)	
	$k=9$	152.9 (8.1)	2.6e-02 (1.5e-04)		465.1 (13.6)	1.0e-02 (1.1e-04)	
	$k=10$	158.2 (9.5)	2.9e-02 (1.7e-04)		480.9 (16.5)	1.1e-02 (1.2e-04)	
FFP	$l=6$	16.6 (0.4)	8.0e-02 (7.3e-04)		17.2 (1.3)	3.2e-02 (5.3e-04)	
	$l=7$	28.5 (0.8)	8.2e-02 (8.4e-04)		28.7 (1.4)	2.7-02 (4.7e-04)	
	$l=8$	58.0 (5.4)	1.9e-02 (5.3e-04)		58.9 (1.9)	2.7e-02 (4.3e-04)	
	$l=9$	100.9 (3.1)	**1.7e-02 (3.5e-04)**	8.9e-29	107.5 (2.9)	2.2e-02 (5.6e-04)	
	$l=10$	196.5 (6.7)	8.9e-02 (8.3e-04)		213.9 (7.3)	**8.8e-04 (3.3e-05)**	2.7e-24
	$l=11$	364.1 (8.1)	1.7e-01 (9.6e-04)		391.2 (12.3)	2.1e-02 (3.3e-04)	
	$l=12$	1724.8 (20.9)	2.8e-01 (9.2e-04)		737.1 (21.1)	6.9e-02 (8.1e-04)	
	$l=13$	1375.9 (36.0)	2.8e-01 (8.6e-04)		1282.2 (35.9)	1.2e-01 (1.2e-03)	
	$l=14$	2480.1 (94.7)	3.2e-01 (8.6e-04)		2039.5 (94.3)	1.5e-01 (1.6e-03)	
	$l=15$	3801.9 (94.6)	3.4e-01 (8.4e-04)		2790.6 (145.2)	1.8e-01 (1.9e-03)	
NW	default	6039.6 (140.7)	-	-	61161.8 (885.0)	-	-

collected from *Yanomani*, the uncontacted Amerindians, consisting of 6,734,572 sequences of 151 bp that cover the V4 hyper-variable region of the 16S rRNA gene. The second dataset (RT988, unpublished data), which covers the V3-V4 regions, was generated from 90 oral plaque samples and contains 4,119,942 sequences of 464-465 bp. Both datasets were generated by Illumina MiSeq. Before the analysis, pre-processing was performed including pair-end joining, quality filtering and length filtering. It is worth noting that since amplicon sequences produced by Illumina MiSeq have fairly stable read lengths, length filtering retained the vast majority of sequences and we did not perform any trimming on the sequences.

3.2 Benchmark Study on Accuracy and Efficiency

When evaluating an alignment-free method, distance estimation accuracy and computational efficiency are two major considerations. Thus, we calculated the mean squared error (MSE) between alignment distances and embedding distances estimated by an alignment-free method, and recorded its computational time. Since it is computationally unfeasible to align all sequence pairs for a dataset, we randomly sampled 2,000 sequences without replacement, and calculated the alignment distances of all 1,999,000 possible sequence pairs. The NW algorithm with the default settings was used for sequence alignment (match score = 5, miss score = -4, gap opening score = -10, gap extending score = -1). To minimize statistical variations, the above sampling process was repeated 10 times. Therefore, we generated a total of 10 testing datasets for each sequence dataset to evaluate the six competing methods.

For kmacs, FFP (V3.19) and Kr (V2.0.2), we used the source code downloaded from their websites. Since kmacs is a extension of ACS, we used the kmacs binary with parameter $k = 0$ as the implementation of the ACS method. For k-mer, we used our own C++ implementation optimized for amplicon sequence data by using sparse k-mer count representation. For k-mer, kmacs and FFP, different parameters can be applied. Since there is no principal way to estimate the optimal parameter, we tested 10 parameters for each method. For the proposed method, we need to train the model for a specific dataset, which can be done offline. With the help of the proposed landmark-based training sample selection algorithm, the number of selected training pairs is trivial compared to the number of all sequence pairs. For both datasets, the number of landmarks was set to 100 and the number of candidates was set to 5,000, resulting in 500,000 sequence pairs for training. The 5,100 sequences used for training account only for $\sim$0.1% of total sequences for both datasets. To prevent information leakage, we also verified that none of the sequences used for training were in testing data. To train the Siamese neural network, we used the Adam optimizer (Kingma and Ba, 2014) and set the learning rate to 1e-4, the number of training epochs to 100, and the embedding dimension to be same as the input sequence length (i.e., 151 for Qiita and 465 for RT988).



(a) SENSE (b) ACS (c) Kr

(d) k-mer ($k = 3$) (e) kmacs ($k = 2$) (f) FFP ($l = 10$)

Fig. 3. Visualization of alignment distances versus estimated distances computed by six alignment-free methods performed on the RT988 dataset.

Table 1 reports the CPU time and MSE results averaged over ten runs for the six methods performed on the Qiita and RT988 datasets. Figs. 2 and 3 plot the estimated distances against the alignment distances for the two datasets, respectively. Due to space limitations, only the best result for each method is reported, and other results are reported in Supplementary Figs. 2 and 3. From the table and figures, we can see that our method achieved the best performance in terms of both accuracy and efficiency. Remarkably, the MSEs of our method are one to two orders of magnitude smaller than other methods. The kmacs method performed a distant second among the six methods in terms of accuracy. However, kmacs runs about 20 times slower and its MSEs are one order of magnitude larger than our method on both datasets. In fact, as we will see shortly, the computational efficiency of our method can be further improved by reducing the embedding dimension while maintaining the same level of accuracy. As a reference, the CPU time of the NW algorithm is also reported in Table 1. The application of our method led to 1,161 fold and 4,497 fold increase in speed for Qiita and RT988, respectively.

We also performed a parameter sensitivity analysis to investigate how the proposed method performs with respect to different filter lengths. We applied our method to the Qiita dataset and reported in Fig. 4 the mean square errors obtained by using various filter lengths ranging from 3 to 8. While the existing methods are quite sensitive to the specific choice of the parameter as shown in Table 1 and the optimal parameter can only be estimated through try-and-error, our method is very robust against different filter lengths. This could be explained by the fact that SENSE is a supervised learning based method that can adjust the weights of the filters automatically.

3.3 Estimating Dimension of Output Embedding Vectors

In the above experiment, we simply set the length of input sequences as the default parameter for the number of nodes in the output layer (i.e., the dimension of output vectors). This parameter is directly related to the model complexity of a constructed neural network, both training and test time, and the size of space required to store embedding vectors if further analysis is needed. In this section, we provide a general guideline for estimating an appropriate number of nodes in the output layer. Our approach is in spirit similar to the elbow method widely used in clustering analysis to estimate the number of clusters (Sugar, 1998; Sun *et al.*, 2017). It was observed that as the number of clusters increases, the data fitting error drops quickly and after some point becomes flattened, suggesting that the model starts to fit random noise (Sugar, 1998). A similar technique is also widely used in principal component analysis to estimate intrinsic data dimension (Tenenbaum *et al.*, 2000). Our constructed neural network model can be considered an encoder of biological sequences. Thus, we posit that we could observe an elbow phenomenon. To demonstrate this, we performed an experiment on the two datasets. Fig. 5 shows the MSE results as a function of various embedding dimensions. *Indeed*, we observed such a phenomenon. When the embedding dimension equals to 80 for Qiita or 130 for RT988, the obtained MSEs are already at the same level as those obtained by using the default parameters. Since the calculation of embedding distances is linear with respect to the embedding dimension and, by reducing the number of nodes in the output layer, the network structure can be much simpler, the CPU times reported in Table 1 were further reduced 2 fold for Qiita and 4 fold for RT988.

3.4 Sample Weighting

In real applications, researchers may be more interested in sequences that are similar. Unlike other alignment-free methods, a unique feature of our method is that it provides a way that allows researchers to associate different costs to different training sequence pairs (see Eq. (3)). To demonstrate this, we performed an experiment on the Qiita dataset and applied the following weight function to the loss function:

$$w(d_a) = \begin{cases} 100 & \text{if } d_a \leq 0.2 \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

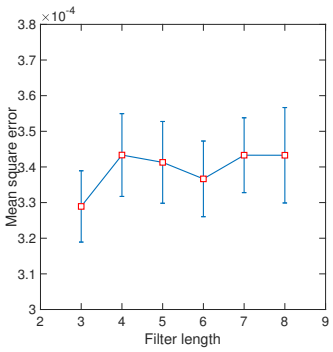
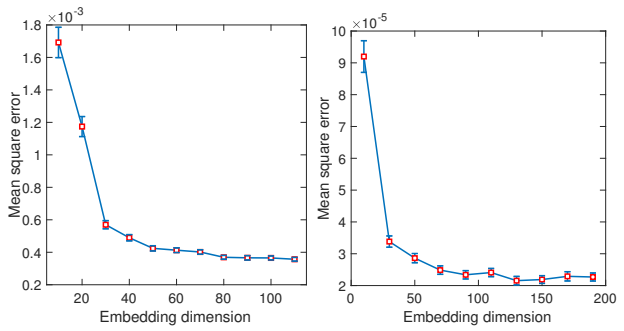


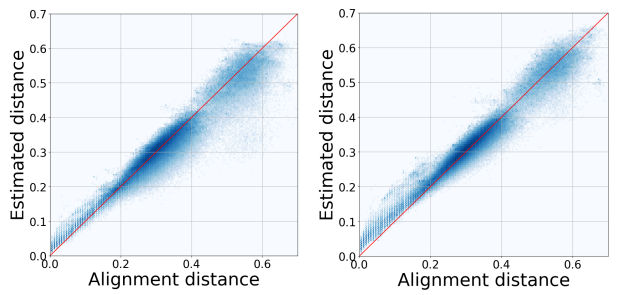
Fig. 4. Results of parameter sensitivity analysis performed on the Qiita dataset. The performance of the proposed method is largely insensitive to a specific choice of filter size.



(a) Qiita (b) RT988

Fig. 5. Estimate embedding dimensions by using the elbow method for (a) Qiita and (b) RT988 datasets.

Fig. 6 shows the estimated distances calculated by SENSE with and without a weight function. We can see that with the weight function the sequence pairs with $d_a \leq 0.2$ are grouped more tightly around the diagonal line. The MSE for those pairs obtained by using the weight function is much smaller than that obtained without the weight function (p-value $\leq$ 2.1e-19, t-test).



(a) w/ weight (b) w/o weight

Fig. 6. Visualization of alignment distances versus estimated distances calculated by SENSE with and without using a weight function.

4 Discussion

In this paper, we developed a novel method for alignment-free sequence comparison using deep convolutional neural networks. We demonstrated that the new method could achieve much more accurate estimation of pairwise distances than other methods. This is expected since SENSE is a supervised learning-based method that learns an embedding function automatically through training while all other methods define sequence similarities based on heuristics. Currently, we used only $\sim$0.1% sequences to train the model. It is possible that the performance of our method can be further improved by using a larger training dataset. Moreover, our method is computationally very efficient, and runs even faster than the k-mer method. Unlike other alignment-free methods, our method is largely insensitive to the specific choice of the parameters, which makes parameter tuning and implementation of our method easy for users.

We should emphasize that our method is different from other machine-learning algorithms used in traditional applications (e.g., face recognition). It is designed mainly for the fast estimation of pairwise distances for large sequence datasets. In our experimental procedure, we trained a model on a small fraction of sequences and then applied the trained model to all the pairs of sequences (though the test data was not used in the training process). However, one would be curious to know whether it generalizes well on independent datasets. To this end, we tested our model on independent datasets and reported the results in Supplementary Table 1. The obtained MSEs are higher than the results reported in Table 1. This is expected because the training data may have a different sample distribution from the test data. Nevertheless, our method still significantly outperformed the best results of all other methods with a large margin. The initial success of our method applied to independent dataset suggests a potential research direction, that is, to develop and publish a trained model (e.g., for 16S V3-V4 sequences) that other researchers can use to process their own datasets. However, we do not think that we are there yet. Currently, our method can only be applied to sequences of roughly equal length. In the future, we plan to use RNN to build a model that can process sequences of various lengths for general applications. Moreover, a large-scale experiment is needed to train a model by using sequences from different datasets.

Alignment-free methods, particularly k-mer, have found numerous applications in sequence analysis. We believe that this work would open a door to develop a range of new methods that do not rely on sequence alignment for large-scale sequence data analysis. By mapping nucleotide sequences to numeric vectors, we also demonstrated that it is possible to mathematically model nucleotide sequences, which may provide a new way to study biological sequences. We plan to perform in-depth analyses to study what type of information in nucleotide sequences has been encoded by a deep learning model.

Funding

This work is supported in part by 1R01AI125982 (YS, RG, JWW), 1R01DE024523 (JWW, RG, YS, WZ), and a grant from Sunstar (RG).

References

Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*.

Bonham-Carter, O., Steele, J., and Bastola, D. (2013). Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, **15**(6), 890–905.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Cai, Y. and Sun, Y. (2011). ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. *Nucleic Acids Research*, **39**(14), e95.

Cai, Y., Zheng, W., Yao, J., Yang, Y., Mai, V., Mao, Q., and Sun, Y. (2017). ESPRIT-Forest: parallel clustering of massive amplicon sequence data in subquadratic time. *PLoS Computational Biology*, **13**(4), e1005518.

Clemente, J. C., Pehrsson, E. C., Blaser, M. J., Sandhu, K., Gao, Z., Wang, B., Magris, M., Hidalgo, G., Contreras, M., Noya-Alarcón, Ó., *et al.* (2015). The microbiome of uncontacted Amerindians. *Science Advances*, **1**(3), e1500183.

Csáji, B. C. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, **24**, 48.

Dos Santos, C. N. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *International Conference on Computational Linguistics*, pages 69–78.

Gao, L. and Qi, J. (2007). Whole genome molecular phylogeny of large dsDNA viruses using composition vector method. *BMC Evolutionary Biology*, **7**(1), 41.

Gao, Y. and Luo, L. (2012). Genome-based phylogeny of dsDNA viruses by a novel alignment-free method. *Gene*, **492**(1), 309–314.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *International Conference on Machine Learning*, pages 1319–1327.

Haubold, B., Pfaffelhuber, P., Domazet-Loso, M., and Wiehe, T. (2009). Estimating mutation distances from unaligned genomes. *Journal of Computational Biology*, **16**(10), 1487–1500.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

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

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–13.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**(4), 541–551.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, **521**, 436–444.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616.

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

Levandowsky, M. and Winter, D. (1971). Distance between sets. *Nature*, **234**(5323), 34–35.

Mao, Q., Zheng, W., Wang, L., Cai, Y., Mai, V., and Sun, Y. (2015). Parallel hierarchical clustering in linearithmic time for large-scale sequence analysis. In *IEEE International Conference on Data Mining*, pages 310–319.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814.

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

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

Song, K., Ren, J., Reinert, G., Deng, M., Waterman, M. S., and Sun, F. (2013). New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, **15**(3), 343–353.

Sugar, C. A. (1998). *Techniques for Clustering and Classification with Applications to Medical Problems*. Ph.D. thesis, Stanford University.

Sun, Y., Cai, Y., Liu, L., Yu, F., Farrell, M. L., McKendree, W., and Farmerie, W. (2009). ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Research*, **37**(10), e76.

Sun, Y., Cai, Y., Mai, V., Farmerie, W., Yu, F., Li, J., and Goodison, S. (2010). Advanced computational algorithms for microbial community analysis using massive 16S rRNA sequence data. *Nucleic Acids Research*, **38**(22), e205.

Sun, Y., Yao, J., Yang, L., Chen, R., Nowak, N. J., and Goodison, S. (2017). Computational approach for deriving cancer progression roadmaps from static sample data. *Nucleic Acids Research*, **45**(9), e69.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**(5500), 2319–2323.

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

Voevodski, K., Balcan, M.-F., Röglin, H., Teng, S.-H., and Xia, Y. (2012). Active clustering of biological sequences. *Journal of Machine Learning Research*, **13**, 203–225.

Wang, Q., Garrity, G. M., Tiedje, J. M., and Cole, J. R. (2007). Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, **73**(16), 5261–5267.

Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, **15**(3), R46.

Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. (2003). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528.

Zheng, W., Mao, Q., Genco, R. J., Wactawski-Wende, J., Buck, M., Cai, Y., and Sun, Y. (2018). A parallel computational framework for ultra-large-scale sequence clustering analysis. *Bioinformatics*, **in press**.

Zielezinski, A., Vinga, S., Almeida, J., and Karlowski, W. M. (2017). Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, **18**(1), 186.

SENSE: Siamese neural network for sequence embedding and alignment-free comparison

Supplementary Data

Wei Zheng[1], Le Yang[1], Robert J. Genco[2,3], Jean Wactawski-Wende[4], Michael Buck[5], Yijun Sun[1,3*]

[1]Department of Computer Science and Engineering
[2]Department of Oral Biology
[3]Department of Microbiology and Immunology
[4]Department of Epidemiology and Environmental Health
[5]Department of Biochemistry
The State University of New York, Buffalo, NY 14203

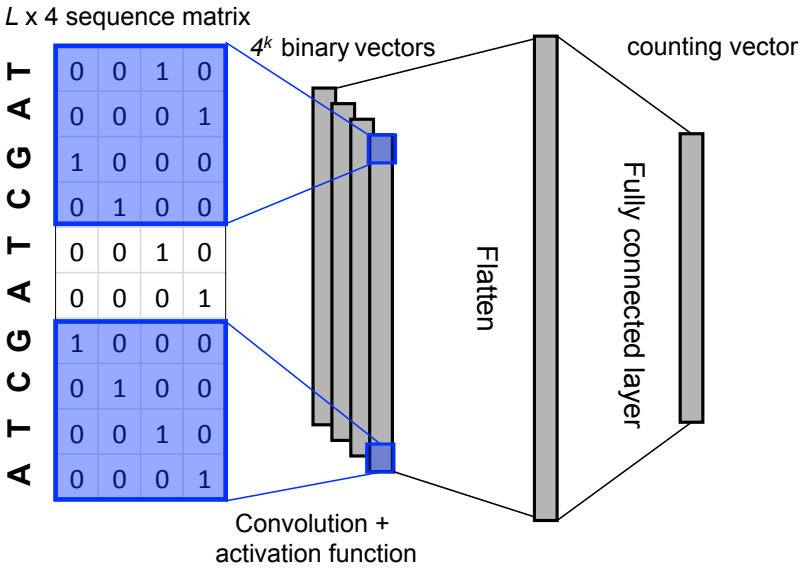
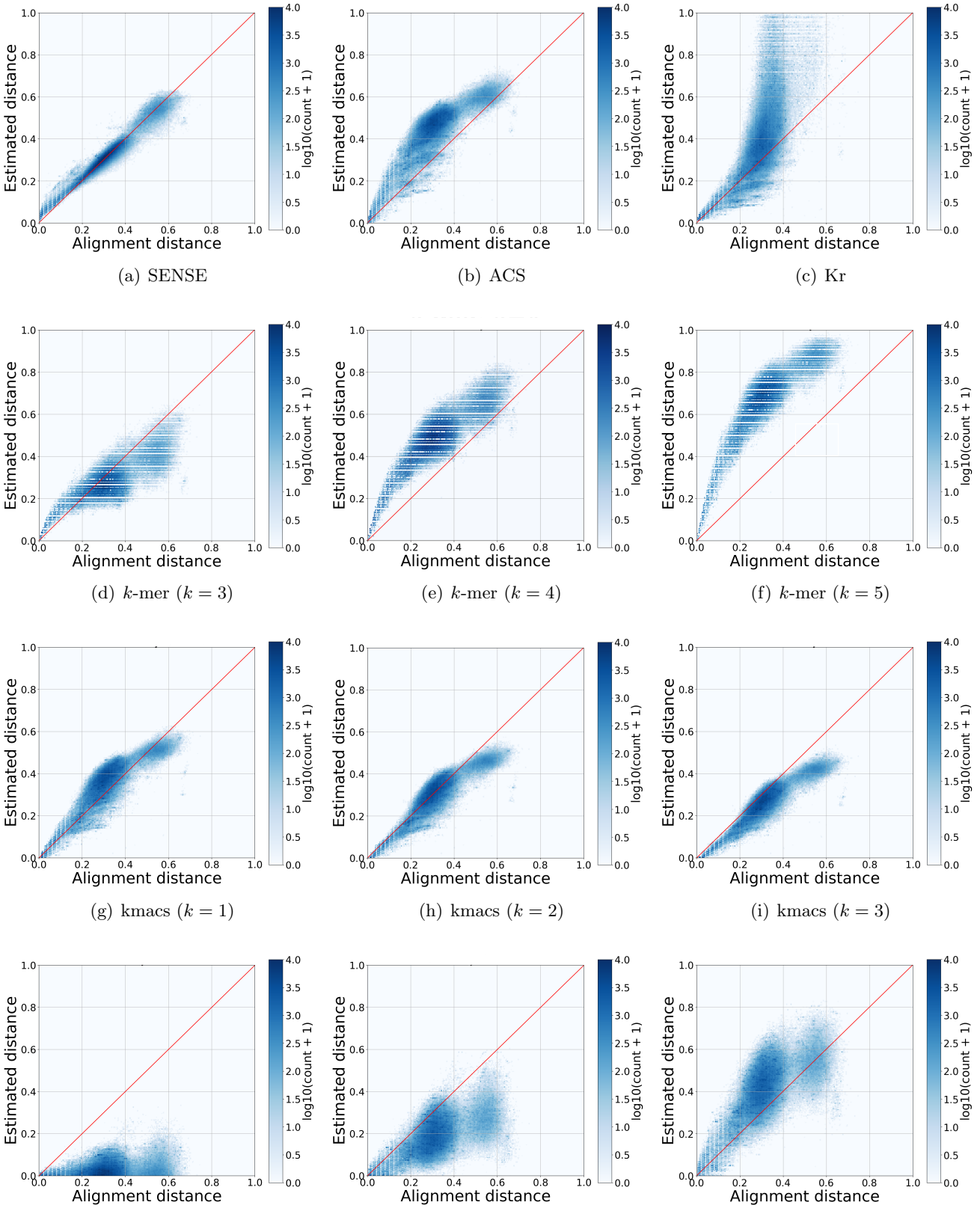


Figure S1: The k-mer method can be implemented as an one-layer convolutional neural network.

*Please address all correspondence to Dr. Yijun Sun (yijunsun@buffalo.edu).



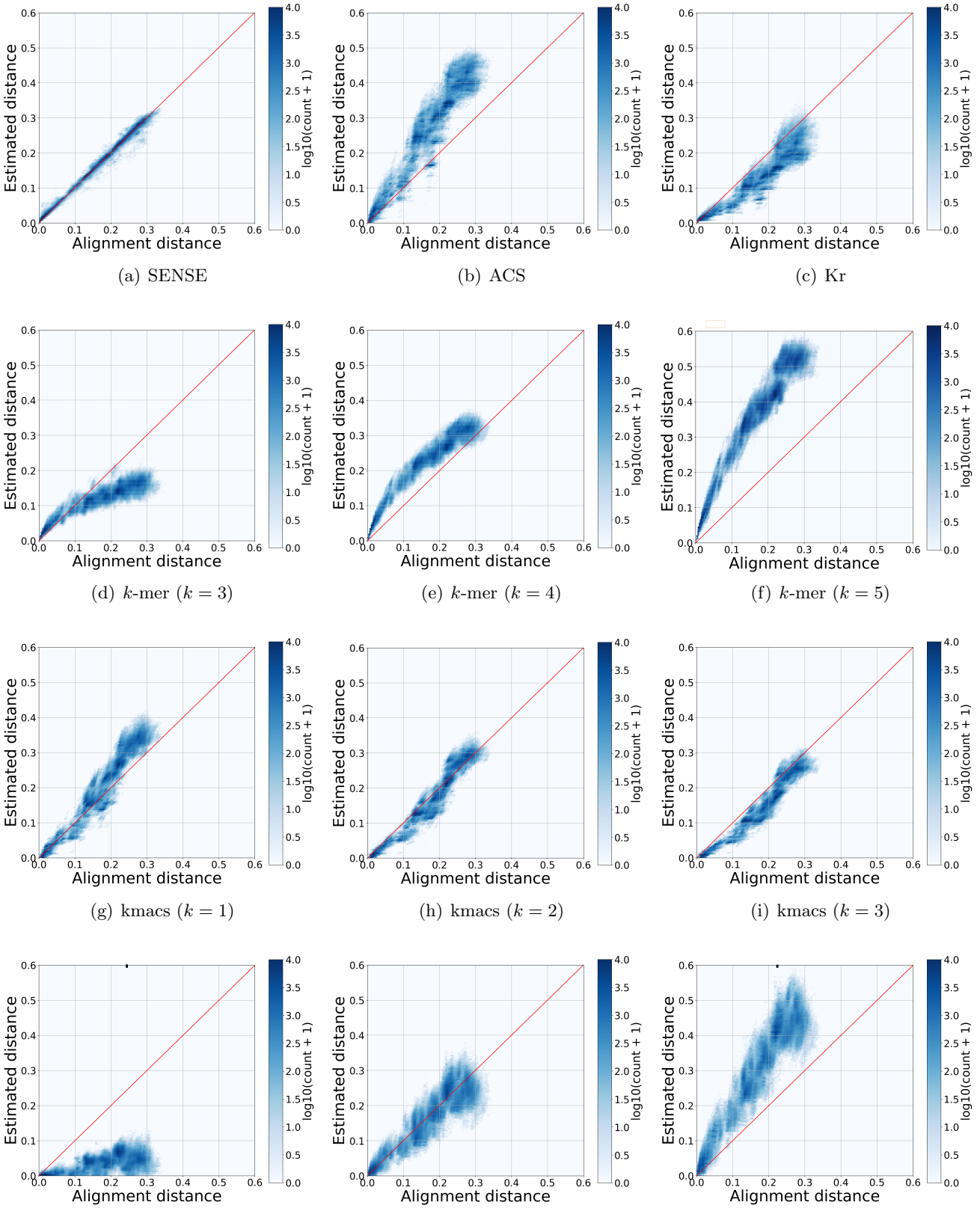
(a) SENSE (b) ACS (c) Kr

(d) k-mer ($k = 3$) (e) k-mer ($k = 4$) (f) k-mer ($k = 5$)

(g) kmacs ($k = 1$) (h) kmacs ($k = 2$) (i) kmacs ($k = 3$)

(j) FFP ($l = 7$) (k) FFP ($l = 8$) (l) FFP ($l = 9$)

Figure S2: Visualization of alignment distances versus estimated distances computed by six alignment-free methods performed on the Qiita dataset. For a method where different parameters can be applied, the top three best results are reported.



(a) SENSE (b) ACS (c) Kr

(d) k-mer ($k = 3$) (e) k-mer ($k = 4$) (f) k-mer ($k = 5$)

(g) kmacs ($k = 1$) (h) kmacs ($k = 2$) (i) kmacs ($k = 3$)

(j) FFP ($l = 9$) (k) FFP ($l = 10$) (l) FFP ($l = 11$)

Figure S3: Visualization of alignment distances versus estimated distances computed by six alignment-free methods performed on the RT988 dataset. For a method where different parameters can be applied, the top three best results are reported.

Table S1: CPU time (in second) and MSE results averaged over ten runs for six methods performed on the Qiita1001 and RT1194 datasets. The SENSE model was trained on the Qiita 10052 and RT988 datasets, respectively. The numbers in parentheses are standard deviations. When different parameters were used for a method, the best result is boldfaced. A p-value was computed by comparing the MSE result of SENSE with the best result of a method.

		Qiita1001			**RT1194**		
Method	**Parameter**	**CPU time**	**MSE**	***p*-value**	**CPU time**	**MSE**	***p*-value**
SENSE	default	4.6 (0.1)	**5.8e-04 (1.2e-05)**	–	11.2 (0.2)	**4.4e-05 (7.7e-06)**	–
ACS	default	56.1 (0.4)	**2.4e-02 (2.3e-04)**	7.6e-35	161.3 (0.8)	**1.1e-02 (1.8e-04)**	2.0e-29
Kr	default	365.8 (5.2)	**5.6e-01 (5.0e-02)**	1.1e-17	652.8 (2.0)	**2.8e-03 (4.2e-05)**	2.1e-28
k-mer	$k = 3$	9.4 (0.4)	**2.6e-03 (7.5e-05)**	2.0e-24	9.0 (0.1)	4.9e-03 (1.7e-04)	
	$k = 4$	14.5 (0.4)	3.8e-02 (2.5e-04)		20.0 (0.2)	**3.1e-03 (4.4e-05)**	2.0e-30
	$k = 5$	17.4 (0.3)	1.4e-01 (5.3e-04)		33.1 (0.6)	3.6e-02 (4.9e-04)	
	$k = 6$	18.2 (0.5)	2.4e-01 (8.2e-04)		40.3 (0.3)	9.3e-02 (1.3e-03)	
	$k = 7$	18.2 (0.5)	3.1e-01 (9.6e-04)		42.7 (0.3)	1.4e-01 (1.8e-03)	
	$k = 8$	18.3 (0.5)	3.5e-01 (1.1e-03)		43.3 (0.4)	1.8e-01 (2.1e-03)	
	$k = 9$	18.1 (0.4)	3.5e-01 (1.1e-03)		43.0 (0.2)	1.8e-01 (2.1e-03)	
	$k = 10$	18.0 (0.4)	3.5e-01 (1.1e-03)		42.9 (0.2)	1.8e-01 (2.1e-03)	
	$k = 11$	17.9 (0.3)	3.5e-01 (1.1e-03)		43.0 (0.6)	1.8e-01 (2.1e-03)	
	$k = 12$	17.7 (0.4)	3.5e-01 (1.1e-03)		42.7 (0.2)	1.8e-01 (2.1e-03)	
kmacs	$k = 1$	84.2 (0.5)	4.5e-03 (6.4e-05)		244.6 (1.35)	1.6e-03 (4.6e-05)	
	$k = 2$	93.1(0.3)	**1.3e-03 (3.4e-05)**	1.1e-22	278.2 (0.7)	**4.2e-04 (2.5e-05)**	9.2e-19
	$k = 3$	101.3 (0.3)	3.2e-03 (6.7e-05)		303.8 (0.8)	1.3e-03 (3.3e-05)	
	$k = 4$	108.5 (0.2)	6.5e-03 (1.0e-04)		327.0 (1.0)	3.0e-03 (3.3e-05)	
	$k = 5$	115.7 (0.3)	1.0e-02 (1.3e-04)		349.6 (0.9)	4.7e-03 (3.6e-05)	
	$k = 6$	122.9 (0.4)	1.4e-02 (1.6e-04)		370.9 (0.9)	6.2e-03 (4.8e-05)	
	$k = 7$	128.4 (0.2)	1.7e-02 (1.8e-04)		387.2 (1.0)	7.8e-03 (7.8e-05)	
	$k = 8$	133.9 (0.6)	2.0e-02 (2.1e-04)		402.2 (1.2)	9.0e-03 (1.0e-04)	
	$k = 9$	139.2 (0.5)	2.3e-02 (2.3e-04)		416.2 (1.3)	1.0e-02 (1.3e-04)	
	$k = 10$	144.0 (0.3)	2.6e-02 (2.4e-04)		428.4 (1.11)	1.1e-02 (1.5e-04)	
FFP	$l = 6$	16.0 (0.4)	5.9e-02 (8.9e-04)		15.9 (0.3)	3.1e-02 (6.4e-04)	
	$l = 7$	27.1 (0.2)	6.6e-02 (8.2e-04)		27.2 (0.5)	2.7-02 (5.6e-04)	
	$l = 8$	53.7 (0.2)	**1.4e-02 (2.6e-04)**	2.0e-29	56.2 (1.0)	2.6e-02 (5.1e-04)	
	$l = 9$	95.7 (0.65)	1.7e-02 (2.8e-04)		102.6 (0.9)	2.2e-02 (5.7e-04)	
	$l = 10$	185.3 (0.5)	8.2e-02 (6.5e-04)		203.0 (1.7)	**9.2e-04 (3.4e-05)**	2.7e-24
	$l = 11$	346.6 (1.4)	1.6e-01 (1.1e-03)		373.2 (2.1)	1.9e-02 (2.2e-04)	
	$l = 12$	682.2 (3.1)	2.2e-01 (1.3e-03)		715.6 (3.6)	6.5e-02 (6.5e-04)	
	$l = 13$	1269.3 (9.0)	2.6e-01 (1.6e-03)		1275.7 (15.0)	1.1e-01 (1.0e-03)	
	$l = 14$	2175.4 (32.1)	2.9e-01 (1.7e-03)		2040.1 (31.9)	1.4e-01 (1.3e-03)	
	$l = 15$	3301.6 (49.3)	3.2e-01 (1.8e-03)		2841.1 (58.2)	1.6e-01 (1.6e-03)	