A deep embedded clustering algorithm for the binning of metagenomic sequences

HUYNH QUANG BAO1,2,*, LE VAN VINH3, AND TRAN VAN HOAI1,2,**

1Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam
2Vietnam National University Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam
3Faculty of Information Technology, HCMC University of Technology and Education, Vo Van Ngan, Ho Chi Minh, Vietnam (e-mail: vinhlv@hcmute.edu.vn)

*baohq1595@gmail.com
**hoai@hcmut.edu.vn

Corresponding author: Le Van Vinh (e-mail: vinhlv@hcmute.edu.vn).

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM), under grant number B2019-20-06.

ABSTRACT The study of metagenomic sequences brings a deep understanding of microbial communities. One of the crucial steps in metagenomic projects is to classify sequences into different organisms, named the binning problem. In the emerging methods for classification, deep learning is a potential technology to be applicable with high accuracy. However, it is well-known that reference databases, which are highly required by deep learning based methods, are not always available. As a result, some existing binning solutions have applied unsupervised learning processes, but utilizing the strength of deep learning in an unsupervised model is still a challenging problem. This work proposes a binning algorithm for metagenomic sequences, called MetaDEC, which applies a deep unsupervised learning approach. By following the two-phase paradigm, the algorithm firstly divides sequences into groups of overlapping sequences. The groups are then classified into clusters using an adversarial deep embedded clustering technique. Experimental results show that MetaDEC achieves competitive performance compared to existing methods on both simulated and real metagenomic data.

INDEX TERMS algorithm, clustering, deep learning, metagenomics, DNA sequence.

I. INTRODUCTION

METAGENOMICS is the study of genetic content collected directly from the environment, bypassing the need for culturing and isolating in laboratories. The field offers opportunities to get a deep inside into microbial communities that are infeasible with traditional methods of single-genome sequencing technologies. There are many fields of applications from metagenomic studies such as biomedical science, biotechnology, energy, and agriculture [1].

Some initial metagenomic projects, e.g. Acid Mine Drainage, and Sargasso Sea are based on Sanger sequencing technology [2], [3]. The technology produces quite long sequences which provide meaningful information for the binning process. Unfortunately, because it requires high costs and is very time-consuming, the technology is impractical for most current projects, which require analyzing a huge amount of data. Thanks to the development of Next-Generation Sequencing approaches such as Illumina, and 454 pyrosequencing [4] which are appropriate for nowadays metagenomic projects. The technologies are able to process millions of reads per run in a short time with low costs. However, one of the limits of the sequencing technologies is that they produce sequences with short lengths, and thus brings research challenges for communities because of the lack of information in short sequences.

Due to the fact that DNA fragments in metagenomic samples are from multiple genomes, one of the major steps in metagenomic projects is to classify sequences into groups of closely related genomes. The step is referred to as binning problem. Binning results are beneficial to reconstructing genomes by assembly approaches or used for analyzing sequences directly from every single genome.

Some binning approaches such as TIPP2 [5], and mOTUs2 [6] use marker genes, e.g. 16S rRNA, recA for profiling metagenomic sequences. The strength of those methods is that they require low computing costs because of not needing
to analyze the whole genomes of organisms. However, the classification quality of the methods is affected by the fact that many different species contain the same marker genes, and others may have a small ratio of 16S rRNA genes [7], [8]. This work focuses on whole-genome sequencing approaches which analyzing all genome sequences of microbial organisms. Those approaches can be classified into groups of supervised, and unsupervised methods.

Supervised methods are based on homology or composition information from an available reference database to support the classification process. MEGAN CE [9], and MEGAN-LR [10] use homology search tools such as DIAMOND [11], or LAST [12] to determine the similarity between input sequences with reference sequences. The algorithms then assign the sequences into groups of known organisms. One of the drawbacks of the methods is that they are very time-consuming. Kraken2 [13] deal with the challenge by extracting long k-mers from sequences and comparing them with reference databases. On the other hand, DeepMicrobes [14] is a composition-based supervised methods that utilizes genomic signatures extracted from sequences to classify the metagenomic data. The method applies a deep learning-based computational framework for its classification task.

Due to the limitation on reference databases, some binning algorithms apply an unsupervised learning process for classifying sequences. MetaCluster 2.0 [15] and MetaCRS [16] aim to cluster metagenomic data of long sequences or contigs. Both of the algorithms use composition features and apply k-means algorithm in their process. Focusing on analyzing short sequences, MetaCluter 5.0 [17] applies a two-round binning approach that classifies effectively samples with low-abundance species. Another binning algorithm for short reads, AbundanceBin [18] is only based on abundance levels of species in metagenome, and thus it does not work well for samples of species with similar abundance levels. BiMeta [19] and MetaProb [20] are other clustering algorithms that utilize sequence overlapping information and k-mer frequency extracted from groups of reads. While BiMeta uses Euclidean distance of k-mer frequencies between data points in its second phase, MetaProb applies probabilistic signatures to get better classification quality. MetaProb 2 [21] is an improvement of MetaProb which requires an assembly task on the group of reads and a graph clustering algorithm for classifying sequences.

Recent studies on unsupervised deep learning-based clustering approach achieve significant improvement [22], [23] and are applied in a few metagenomic binning algorithms. Among them, the binning algorithm of Isis et al [24] uses an autoencoder architecture to compress composition-based representation of sequences and applied k-Means++ method on compressed representation to classify data. On the other hand, VAMB algorithm [25] only focuses on analyzing metagenomic contigs. It uses a variational autoencoder architecture to compress data into a latent posterior distribution (multivariate Gaussian distribution) in the clustering process.

This study proposes a novel unsupervised binning method for metagenomic sequences called MetaDEC (i.e., Metagenomic binning with Deep Embedded Clustering). The proposed approach follows the paradigm of two phases as used by previous studies [19], [20] in which the first phase does preprocessing work to groups sequences using the sequence overlapping information. However, different from previous works, MetaDEc then uses ADEC [23] - a novel unsupervised deep learning approach - to classify the sequence groups in its second phase. The method utilizes an autoencoder architecture with latent space interpolation factor to learn the underlying representation of data and optimize the learned representation toward the clustering objective.

The next section presents the details of the proposed method. The Experiments and Results section shows the strength of MetaDEC comparing with available binning approaches. Some conclusions are presented in the final section.

II. METHODS

In order to overcome the lack of phylogenetic information in short sequences, the proposed method applies a two-phase paradigm in the clustering process (Fig. 1). Phase 1 does preprocessing work to build groups of sequences using overlapping information between them. Clustering features are then extracted from the sequence groups. Phase 2 uses the features to continue classifying the groups into clusters of close organisms.

1) Phase 1: Grouping sequences and building seeds

Derived from the work in [19], the phase classifies reads that share sufficient long l-mer into the same groups and builds group representative. Based on an observation that the l-mers are unique in genomes [17], [26]. MetaDEC firstly builds a graph whose vertices are reads, and each edge is the connection between two reads that have sufficient substring overlapping. While previous methods [19], [20] used a greedy approach to build groups, this work applies a multi-level partitioning algorithm [27] to create a coarsened graph and acquires groups from connected components.

An observation in [19] revealed that genomics signatures of k-mer nucleotide frequency are also preserved in a group of non-overlapping short reads as in long sequences. Thus, MetaDEC selects a subgroup of non-overlapping reads in each group, called seed, as a representative of the group. The technique does not only reduce noises in features extracted from sequence groups that have unbalanced coverage but also saves computation costs [19]. Next, k-mer frequency distribution of each seed is computed as follows.

Given $S = \{r_1, r_2, \ldots, r_n\}$ is a seed, where $n$ is the number of reads in seed $S$. Let $|r_i|, i \in [1..n]$ be the length of read $r_i$. In order to find k-mers of each read, MetaDEc uses a sliding window method with a window size of $k$. There is $|r_i| - k + 1$ k-mers in each read. Thus, the total number of k-mer of seed $S$, denoted $|S|$, is $\sum_{i=1}^{n}(|r_i| - k + 1)$. 

In other words, there are at most $4^k$ different contents of $k$-mers because each $k$-mer is a combination of 4 kinds of nucleotides (A, T, G, C corresponding to Adenine, Thymine, Cytosine, Guanine, respectively). However, either of the DNA strands can be obtained from their reversed complement strands (e.g., ATTT - TAAA, GCGC - CGCG). The frequency of a $k$-mer and its reversed complement are the same. As a result, the number of total $k$-mer values could be reduced by half, from $4^k$ to $4^k/2$ if $k$ is odd, $(4^k + 4^k/2)/2$ if $k$ is even. Studies in [15], [19], [28] showed that value $k = 4$ in computing $k$-mer frequency is the best choice for extracting compositional features from DNA sequences or contigs. Thus, we also choose $k = 4$ in this work. It means that there are 136 $k$-mers in total.

Let $f^S = \{f^S_1, f^S_2, ..., f^S_{136}\}$ is the set of $k$-mer frequency representation of seed $S$. $f^S$ is normalized by dividing each entry by $|S|$. It is then normalized a second time into a normal distribution $x^S$ with mean $\mu = 0$ and variance $\sigma = 1$. As a result, the final representation of seed $S$ is $x^S = \{x^S_1, x^S_2, ..., x^S_{136}\}$, in which each $x^S_i$ is in the range of $[-1, 1]$.

### A. PHASE 2: CLASSIFY OF SEQUENCE GROUPS USING DEEP CLUSTERING

Given $n$ groups of sequences which are represented by a set of $n$ normalized frequencies of seeds constructed in phase 1 $X = \{x^{S_1}, x^{S_2}, ..., x^{S_n}\}$, in which $S_i, i \in [1..n]$ be seeds of the groups. In this phase, MetaDEC classifies $X$ into $m$ clusters with $m$ centroids $C = \{c_1, c_2, ..., c_m\}$ applying a deep clustering method called ADEC (Adversarial Deep Embedded Clustering) [23]. The phase consists of two steps: Cluster initialization, and Clustering optimization.

1) Step 1: Clusters initialization

The first step aims to find a sufficient initialization of clusters (presented in Fig. 1 and Algorithm 1). Instead of using similarity measurement directly on $X$, the proposed method uses an autoencoder to learn the latent space representation of $X$. In detail, the autoencoder composes of two parts, encoder, and decoder. The encoder is a non-linear mapping function $E_{\theta} : X \rightarrow Z$ that transforms $X$ space into a smaller dimensionality latent feature space $Z = \{z^{S_1}, z^{S_2}, ..., z^{S_n}\}$, in which each $z^{S_i}$ corresponds to encoded $x^{S_i}, i \in [1..n]$. The decoder is another non-linear mapping function $D_{\phi} : Z \rightarrow \hat{X}$ that transforms latent space $Z$ back to the original data space of $k$-mer frequency representation. In our work, both encoder and decoder are modeled by neural networks. MetaDEC also adds interpolation factor on latent space based on a framework proposed from ACAI [29] by using another neural network called critic $C_{\psi}$ to improve the quality of latent space. The information from the critic is used as a regularization term in the autoencoder optimization process.

In this step, autoencoder (encoder $E_{\theta}$, decoder $D_{\phi}$) and critic $C_{\psi}$ are trained in predefined iterations. Each iteration updates $C_{\psi}$ using a loss function $L_C$ (1) and $E_{\theta}$, $D_{\phi}$ using a loss function $L_{E,D}$ (2).

\[
L_C = ||C_{\psi}(\hat{x}^S_\alpha) - \alpha||^2 + ||C_{\psi}(\gamma x^{S_1} + (1 - \gamma)D_{\phi}(E_{\theta}(x^{S_1})))||^2
\]  

\[
L_{E,D} = ||x^{S_1} - D_{\phi}(E_{\theta}(x^{S_1}))||^2 + \lambda||C_{\psi}(\hat{x}^S_\alpha)||^2
\]

in which $\hat{x}^S_\alpha = D_{\phi}(\alpha E_{\theta}(x^{S_1}) + (1 - \alpha)E_{\theta}(x^{S_2}))$, $x^{S_1}, x^{S_2}$ are randomly picked data points in $X$ and $\alpha, \lambda$ are randomly sampled in range $[0, 1]$. It is noted that instead of using square root function for each term in $L_C$ and $L_{E,D}$ as the work.
in [23], MetaDEC uses square function for a better model training.

After the optimization of the autoencoder and critic, latent space \( Z = E_\theta(X) \) is computed, and centroids \( C \) are initialized by applying \( k \)-means algorithm on \( Z \).

2) Step 2: Clustering optimization

In this step, MetaDEC continues optimizing the clustering result from step 1. The process performs two sub-steps iteratively, which computes soft clustering assignment and learns from high confidence assignment. It runs until convergence, or a threshold met.

Firstly, soft cluster assignment for latent space \( Z \) is computed. This study uses the Student’s t-distribution in calculating the distance from \( z^{S_i}, i \in [1..n] \) and cluster’s centroid \( c_j, j \in [1..m] \). The probability \( q_{ij} \in Q \) that data point \( i^{th} \) belongs to cluster \( j^{th} \) is computed as the following formulation.

\[
q_{ij} = \frac{(1 + \| z_i - c_j \|^2/\eta)^{-\eta/2}}{\sum_{j' = 1}^m (1 + \| z_i - c_{j'} \|^2/\eta)^{-\eta/2}}
\]

in which, \( z^{S_i} = E_\theta(x^{S_i}) \in Z, \eta \) is degree of freedom in t-student distribution. Assignment of data point \( i \) is computed as \( y_{i}^{pred} = \arg\max_{j}(q_{ij}) \).

Secondly, auxiliary target distribution \( P = p_{ij}, i \in [1..n], j \in [1..m] \), is computed by deriving from high confidence assignment as follows.

\[
p_{ij} = \frac{q_{ij}^2 / freq_{ij}}{\sum_{j=1}^m q_{ij}^2 / freq_{ij}}
\]

in which \( freq_{ij'} = \sum_{i} q_{ij'} \) is the frequency of soft assignment. An objective here is to minimize the difference between soft cluster assignment distribution \( Q \) and auxiliary target distribution \( P \). MetaDEC then minimizes Kullback-Leibler divergence (KLD) between \( P \) and \( Q \) to optimize \( E_\theta \) and clusters’ centroids \( c_i \in C \).

Beside optimizing KLD loss function, MetaDEC adds a neural network into the clustering optimization step, called discriminator \( G_\omega \), to help improve the quality of latent space produced by encoder \( E_\theta \). The discriminator network is optimized to be able to distinguish real groups and reconstructed groups by minimizing loss function \( L_G \) (equation 5).

The information produced by discriminator \( G_\omega \) is used as a regularization term in optimizing \( E_\theta \) and clusters’ centroids \( C \). The regularization term penalizes the encoder when it generates meaningless latent space for decoding. Equation 6 describes the final loss function \( L_{cls} \) used for optimizing \( E_\theta \) and clusters’ centroids \( C \).

The proposed method keeps optimizing decoder \( D_\phi \) in the training process. The decoder in this step plays a role as a monitor for assuring the quality of the latent space. Thus, it needs to be trained to catch up with the changes from encoder \( E_\theta \) by optimizing reconstruction loss \( L_D \) in (7).

\[
L_G = E_{x \sim p(x)}[log(G_\omega(x))] + E_{x \sim p(x)}[log(1 - G_\omega(D_\phi(E_\theta(x))))]
\]

\[
L_{cls} = KLD(P||Q) + E_{x \sim p(x)}[log(1 - G_\omega(D_\phi(E_\theta(x)))]
\]

\[
L_D = ||x - D_\phi(x)||^2_2
\]

Algorithm 2 presents the details of clustering optimization step. It firstly does pretraining discriminator before finding assignment results for elements in \( X \) to clusters.

B. PERFORMANCE METRICS

The quality performance of binning algorithms are evaluated using three metrics precision, recall and F-measure (also used in [17], [19], [20]). While precision measures the ratio of reads classified into a cluster that comes from the same species, recall presents the percentage of reads that belong to the same species are clustered into the same clusters. Because each of the two metrics cannot fully reflect the performance of a binning algorithm, F-measure is used as a harmonic metric between them. The three metrics are defined as follows.

\[
\text{precision} = \frac{\sum_{i=1}^k \text{max}_j A_{ij}}{\sum_{i=1}^k \sum_{j=1}^m A_{ij}}
\]

\[
\text{recall} = \frac{\sum_{i=1}^k \sum_{j=1}^m \text{max}_i A_{ij}}{\sum_{i=1}^k \sum_{j=1}^m A_{ij} + \neq \text{unassigned reads}}
\]

\[
F - \text{measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

in which \( A_{ij} \) is the number of reads that come from species \( j \) classified into cluster \( i \), \( m \) is the number of species in the
Algorithm 2: Cluster optimization

**Input:** Input groups: \( X \); maximum clustering iterations: \( \text{max\_iter} \); auxiliary iterations: \( \text{aux\_iter} \); maximum iterations for pretraining discriminator: \( \text{max\_iter\_dis} \); discriminator learning rate: \( \delta_{\text{dis}} \); cluster learning rate: \( \delta \); clusters’ centroids: \( C \); target update iterations: \( \text{tar\_iter} \); tolerance threshold: \( \text{tol} \).

**Output:** Assignment results to element in \( X \) to specific cluster.

1. **Pretraining discriminator**
   1.1. **for** \( i \leftarrow 1 \) **to** \( \text{max\_iter\_dis} \) **do**
   1.2. Compute \( L_G \) using (3)
   1.3. Update parameter
   1.4. \( \omega \leftarrow \omega - \delta \Delta L_G \)
   1.5. **end for**

2. **Clustering optimization**
   2.1. **for** \( i \leftarrow 1 \) **to** \( \text{max\_iter} \) **do**
   2.2. **if** \( \text{tar\_iter} == 0 \) **then**
   2.3. Compute \( Q \) using (3)
   2.4. **Save** \( y_{\text{pred}} \) **result of previous step**
   2.5. \( y_{\text{pred\_old}} \leftarrow y_{\text{pred}} \)
   2.6. Compute \( y_{\text{pred}} \)
   2.7. Compute \( P \) using (4)
   2.8. **if** \( \frac{1}{3} \sum (y_{\text{pred}} \neq y_{\text{pred\_old}}) < \text{tol} \) **then**
   2.9. **break**
   2.10. **end if**
   2.11. **end if**
   2.12. Compute \( L_G, L_D, L_{\text{cls}} \) using (5), (6), (7)
   2.13. **if** \( \% \text{aux\_iter} <= (\text{aux\_iter}/2) \) **then**
   2.14. \( \phi \leftarrow \phi - \delta \Delta L_D \)
   2.15. **else**
   2.16. \( \theta \leftarrow \theta - \delta \Delta L_{\text{cls}}; \phi \leftarrow \phi - \delta \Delta L_D \)
   2.17. \( \omega \leftarrow \omega + \delta \Delta L_G; C \leftarrow C - \delta \Delta L_{\text{cls}} \)
   2.18. **end if**
   2.19. **end for**

**B. EXPERIMENTAL SETTINGS**

In the experiment, we use a specific machine for each phase of the proposed algorithm. Phase 1 is run on a machine with 500GB memory and Intel(R) Xeon(R) Gold 6152 processor. Phase 2, which includes the process of training neural networks, is run on a machine of NVIDIA Tesla P100 16GB GPU to accelerate the training time.

In the first phase of MetaDEC, two sequences are considered to be overlapped with each other if they share at least \( n_{\text{olp}} \) overlapping l-mer substrings. This work empirically set \( n_{\text{olp}} = 5 \) for short sequence datasets and \( n_{\text{olp}} = 45 \) for long sequence datasets. Besides, the value of \( l \) of l-mer to determine overlapping sequences in phase 1 of the proposed algorithm is set to 30 empirically.

The second phase of MetaDEC is performed with four types of architecture including Tiny, Small, Large, XLarge which are presented in Appendix B. Experiments for simulated datasets in this work uses Small architecture.

In the other hands, the step of cluster initialization trains autoencoder and critic parts for 2000 iterations using Adam optimizer with a fixed learning rate at 0.001. Besides, in the cluster optimization step, discriminator is trained using the same optimizer with 200 iterations. The step continues to train the model using Adam optimizer with learning rate at 0.0001 with epsilon value 10e-8. The training will be stopped if the ratio of changing clustered data points after two consecutive iterations less than a threshold \( \text{tol} \) of 0.0001% or reaching the maximum number of 300 iterations. In addition, the degree of freedom \( \eta \) in t-student distribution used in step 2 of the phase is set as 1.

**C. RESULTS ON SIMULATED DATASETS**

1) Results on short read datasets

The proposed method is firstly compared with BiMeta, MetaCluster 5.0 on short-read datasets from S1 to S10. The bar charts in Fig. 2 present the precision and recall of the binning algorithms. It can be seen from the chart that MetaDEC gets higher precision values than BiMeta and MetaCluster 5.0 on 9/10 datasets. Besides, there are 6/10 cases in which the proposed method returns better recall values than the two remaining algorithms.

Furthermore, overall F-measures of the three methods are presented in Table 1. The table shows that MetaDEC outperforms both BiMeta and MetaCluster 5.0 for 6/10 samples. On average, MetaDEC gets higher 3.79% and 12.91% compared with the two remaining methods, respectively.

Considering the ability to classify metagenomes of different abundance levels, MetaDEC is compared with BiMeta and AbundaceBin on datasets from L1 to L6. The datasets contain two species of *Eubacterium eligens* and *Lactobacillus amylovorus*, and have different abundance ratios. The
bar chart in Fig. 3 shows that MetaDEC achieves higher F-
measure than both BiMeta and AbundanceBin on all samples
of low-abundance ratios (Datasets from L1 to L4 with ratios
of 1:1, 1:2, 1:3, 1:4, respectively). Especially, MetaDEC
outperforms BiMeta for all cases. Because the strength of
AbundanceBin is based on abundance ratios of species in
datasets to classify sequences, it gets better results than
BiMeta and MetaDEC for samples L5, and L6 (with highly
different abundance levels).

2) Results on long read datasets
The proposed method is also compared with MetaCluster 2.0
and BiMeta on long sequence datasets from R1 to R9. Table 2
shows that MetaDEC outperforms the two algorithms for 8/9
cases. The proposed method gets 3.44% and 11.44% higher
F-measure on average than BiMeta and MetaCluster 2.0 on
the samples, respectively. For more details, bar charts in Fig.
4 show that MetaDEC returns higher precision values than
both the algorithms on 8/9 samples. Besides, there are 6/9
cases that the proposed method achieves better recall values
than MetaCluster 2.0 and BiMeta.

D. RESULTS ON A REAL DATASET
The proposed method is also tested on a real meta-genome. It
is compared with BiMeta on the dataset of AMD. This dataset
is revealed in previous studies that it contains five dominant
species, including Ferroplasma sp. Type II, Leptospirillum
sp. Group II, Leptospirillum sp. Group III, Ferroplasma aci-
$\text{MetaCluster}$ 2.0, $\text{BiMeta}$ [19], and $\text{MetaDEC}$ on datasets from R1 to R9.

Darmanus Type I, and Thermoplasmatales archaeon Gpl with abundance ratios of 5:5:1:1:1, respectively [2]. In order to assess binning results, assembled scaffolds of the five species are downloaded from NCBI. Reads from each cluster in the output results are aligned with the scaffolds using the BLAST tool to calculate roughly classification accuracy.

The dataset is tested using two network architectures of Small and Large presented in Appendix B. Experimental results show that $\text{MetaDEC}$ achieves F-measure values of 68.85% and 70.1% when architecture Small and Large applied, respectively. The results are very comparative to $\text{BiMeta}$ which was reported that it got a F-measure value of 68.3 on the dataset [19].

![FIGURE 4. Precisions and recalls of $\text{MetaCluster}$ 2.0 [19], $\text{BiMeta}$ [19], and $\text{MetaDEC}$ on datasets from R1 to R9.](image)

**E. NETWORK EVALUATION**

The section evaluates the performance and running time of $\text{MetaDEC}$ on different sizes of model architectures. The proposed algorithm is performed on four architectures Tiny, Small, Large, and Xlarge presented in Appendix B on S5 and R3 datasets. It is noted that the architectures are assigned with different numbers of layers (Tiny is 3 layers, Small is 4 layers, Large is 5 layers, and Xlarge is 7 layers). Line chart in Fig. 5 shows that $\text{MetaDEC}$ reaches the highest F-measures with architecture Large on sample S5 and with architecture Small on sample R3. However, it gets the lowest F-measures with architecture Xlarge on both datasets. Furthermore, with the shallow network Tiny, $\text{MetaDEC}$ also returns lower F-measures on the datasets compared with architectures of Small and Large. The results reveal that if a model is too shallow (e.g. Tiny with 3 layers), it is not able to learn meaningful latent space that has the ability to describe original data space. In contrast, a model that is too deep (e.g. Xlarge with 7 layers) is prone to overfitting the data instead of extracting meaningful features from it. In both cases, latent spaces learned from these models would not result in a good clustering performance.

As can be seen from the visualization of latent spaces from results of cluster initialization for the both datasets (section

![FIGURE 5. F-measures of $\text{MetaDEC}$ on dataset S5 and R3 with different learning network architectures.](image)

![FIGURE 6. Running time of $\text{MetaDEC}$ on dataset S5 and R3 with different learning network architectures.](image)
II-A1 in Phase 2 of MetaDEC) presented in Appendix C, latent spaces of species from Tiny and Xlarge models tend to be mixed together. It can result in poor centroid initialization. In contrast, latent spaces of Small and Large models give a more convex shape for each specie, and result in a good centroid initialization.

In the aspect of processing time, the line chart in Fig. 6 shows that the running time of MetaDEC is proportional to the complexity level of model architectures. It can be understood because the larger size of the model is used, the higher computational resources are required.

Furthermore, this work also evaluates the performance of the case of using neural network with zero layer (by applying $k$-means directly on $k$-mer frequency representation). The experiment result is compared with MetaDEC using Small architecture on all of the simulated datasets. The chart in Fig. 7 shows that MetaDEC performs better than the case merely applying $k$-means on average. It means that applying ADEC method providing promising classification quality on metagenomic data.

![FIGURE 7. F-measure of the case with zero layer neural network and MetaDEC with architecture neural network Small. The F-measure values are the average F-measure on datasets S1-S10, L1-L6, R1-R9, respectively.](image)

**IV. CONCLUSION**

The complexity of the microbial community requires in-depth exploration strategies to yield practical benefits. This study takes advantage of deep learning techniques to effectively solve the binning problem of metagenomic data. The proposed method employs an adversarial deep embedding approach to automatically learn a latent space of $k$-mer frequency representation of sequence groups, and is demonstrated to outperform state-of-art binning algorithms. In future works, it is worth extending our method to estimate the number of potential clusters in order to be more suitable in the binning context. Besides, a deeper investigation to find a better measuring function for soft cluster assignment in the second phase of MetaDEC is one of the potential research directions. Source codes and datasets used in this work can be downloaded from [https://bioinfolab.fit.hcmute.edu.vn/MetaDEC](https://bioinfolab.fit.hcmute.edu.vn/MetaDEC).

**REFERENCES**

[1] National Research Council (US) Committee on Metagenomics: Challenges and Functional Applications, “Why Metagenomics?,” *The New Science of Metagenomics: Revealing the Secrets of Our Microbial Planet*. Washington (DC), USA: NAP, 2007.

[2] G. W. Tyson, J. Chapman, P. Hugenholtz, E. E. Allen, R. J. Ram, P. M. Richardson, V. V. Solovey, E. M. Rubin, D. S. Rohksar, and J. F. Banfield, “Community structure and metabolism through reconstruction of microbial genomes from the environment,” *Nature*, vol. 428, no. 6978, pp. 37–43, 2004.

[3] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, W. Nelson, W. Nelson et al., “Environmental genome shotgun sequencing of the sargasso sea,” *Science*, vol. 304, no. 5657, pp. 66–74, 2004.

[4] S. Ji and J. H., “Next-generation dna sequencing,” *Nat Biotechnol.*, vol. 26, no. 10, pp. 1135–1145, 2008.

[5] N. Shah, E. K. Molloy, M. Pop, and T. Warnow, “Tipp2: metagenomic taxonomic profiling using phylogenetic markers,” *Bioinformatics*, vol. 37, no. 13, pp. 1839–1845, 2021.

[6] A. Milanese, D. R. Mende, L. Paoli, G. Salazar, H.-J. Ruscheweyh, M. Cuenca, P. Hingamp, R. Alves, P. I. Costea, L. P. Coelho et al., “Microbial abundance, activity and population genomic profiling with motus2,” *Nat Commun.*, vol. 10, no. 1, pp. 1–11, 2019.

[7] C. Case, H. Boucher, L. Yan and D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” *Nature Methods*, vol. 12, no. 1, pp. 59–60, 2015.

[8] S. M. Kielbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith, “Adaptive seeds tame genome sequence comparison,” *Genome Res.*, vol. 21, no. 3, pp. 487–493, 2011.

[9] D. E. Wood, J. Lu, and B. Langmead, “Improved metagenomic analysis with Kraken 2,” *Genome Biol.*, vol. 20, no. 1, pp. 1–13, 2019.

[10] Q. Liang, P. W. Bible, Y. Liu, B. Zou, and L. Wei, “Deepmicrobes: taxonomic classification for metagenomics with deep learning,” *NAR Genom Bioinform.*, vol. 2, no. 1, pp. lqaa009, 2020.

[11] B. Yang, Y. P., H. C. Leung, S. Yiu, J. Qin, R. Li, and F. Y. Chin, “Metacluster: Unsupervised binning of environmental genomic fragments and taxonomic annotation,” in *ACM-BCB 2010*, New York, NY, USA, 2010, pp. 170–179.

[12] Z. Jiang, X. Li, and L. Guo, “TMetacs: unsupervised clustering of contigs with the recursive strategy of reducing metagenomic dataset’s complexity,” *BMC Bioinform.*, vol. 22, no. 12, pp. 1–17, 2022.

[13] W. Wang, H. C., Leung, S. Yiu, and F. Y. Chin, “Metacluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample,” *Bioinformatics*, vol. 18, no. 28, pp. i356–i362, 2012.

[14] Y. Wu and Y. Ye, “A novel abundance-based algorithm for binning metagenomic sequences using l-tuples,” *J. Comput Biol.*, vol. 18, no. 3, pp. 523–534, 2011.

[15] L. V. Vinh, T. V. Lang, L. T. Binh, and T. V. Hoai, “A two-phase binning algorithm using l-mer frequency on genomes of non-overlapping reads,” *Algorithms Mol Biol.*, vol. 10, no. 1, pp. 2, 2015.

[16] S. Girott, C. Pizzi, and M. Comin, “Metaprob: accurate metagenomic reads binning based on probabilistic sequence signatures,” *Bioinformatics*, vol. 32, no. 17, pp. i567–i575, 2016.

[17] F. Andreacce, C. Pizzi, and M. Comin, “Metaprob 2: metagenomic reads binning based on assembly using minimizers and k-mers statistics,” *Comput Biol.*, vol. 28, no. 11, pp. 1052–1062, 2021.

[18] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” *34th Int. Conf. Mach. Learn. ICML 2017*, vol. 37, pp. 478–487, 2017.
Bao et al.: A deep embedded clustering algorithm for the binning of metagenomic sequences

[23] N. Mrabah, M. Bouguessa and R. Ksantini, “Adversarial Deep Embedded Clustering: On a Better Trade-off Between Feature Randomness and Feature Drift,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 4, pp. 1603–1617, 2022, doi: 10.1109/TKDE.2020.2997772.

[24] I. Bonet, A. Pena, C. Lochmüller, A. Patino, and M. Gongora, “Deep clustering for metagenomics,” CIBB, Bergamo, Italy, 2019, pp. 335-347.

[25] Nissen, J.N., Johansen, J., Allesøe, R.L. et al., “Improved metagenome binning and assembly using deep variational autoencoders,” Nat Biotechnol., vol. 39, no. 5, pp. 555–560, 2021, doi: 10.1038/s41587-020-00777-4.

[26] Y. Wang, H. C. Leung, S. Yiu, and F. Y. Chin, “Metacluster 4.0: a novel binning algorithm for ngs reads and huge number of species,” J. Comput Biol., vol. 1, no. 2, 2012.

[27] G. Karypis and V. Kumar, “Metis - unstructured graph partitioning and sparse matrix ordering system, version 2.0,” 1995.

[28] B. Chor, D. Horn, N. Goldman, Y. Levy, and T. Massingham, “Genomic dna k-mer spectra: models and modality,” Genome Biol., vol. 10, no. 10, pp. 1–10, 2009.

[29] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, “Understanding and improving interpolation in autoencoders via an adversarial regularizer,” ICLR, New Orleans, USA, 2019.

HUYNH QUANG BAO received the B.S. degree in computer engineering from HCMC University of Technology (Vietnam National University, Ho Chi Minh City), in 2013. He is currently pursuing M.S degree in computer science at HCMC University of Technology (Vietnam National University, Ho Chi Minh City). His research interest includes deep learning and metagenomic binning.

LE VAN VINH received the bachelor’s degree in Information Technology, and MSc degree in Computer Science from University of Science (Vietnam National University, Ho Chi Minh City) in 2005 and 2009, respectively. He received the PhD degree in Computer Science from HCMC University of Technology (Vietnam National University, Ho Chi Minh City) in 2017. Currently, he is working at the faculty of Information Technology, HCMC University of Technology and Education, Vietnam. His research interests include Bioinformatics, High-performance computing, Data science, and Deep learning.

TRAN VAN HOAI is an associate professor at HCMC University of Technology (Vietnam National University, Ho Chi Minh City). He received his PhD degree of applied mathematics from Heidelberg University in 2005. His theoretical background is combinatorial optimization. After graduation, he has initiated and joined different projects to apply his knowledge on optimization to various fields. His research interests are metagenomic binning in bioinformatics, convexity in computational geometry and practical optimization problems.
APPENDIX A DATASETS

TABLE 3. Description of simulated datasets

| Samples  | No. of species | Phylogenetic distance | Ratio  | No. of reads |
|----------|----------------|-----------------------|--------|--------------|
| S1       | 2              | Species               | 1:1    | 96367        |
| S2       | 2              | Species               | 1:1    | 195339       |
| S3       | 2              | Order                 | 1:1    | 338725       |
| S4       | 2              | Phylum                | 1:1    | 375302       |
| S5       | 3              | Species and Family    | 1:1:1  | 325400       |
| S6       | 3              | Phylum and Kingdom    | 3:2:1  | 713388       |
| S7       | 5              | Order, Order, Genus, Order | 1:1:4:4 | 1653550    |
| S8       | 5              | Genus, Order, Order, Order | 3:5:9:11 | 456224     |
| S9       | 15             | Various distances     |        | 2234168      |
| S10      | 30             | Various distances     |        | 4990632      |

| Samples  | No. of species | Phylogenetic distance | Ratio  | No. of reads |
|----------|----------------|-----------------------|--------|--------------|
| L1       | 2              | Class                 | 1:1    | 176688       |
| L2       | 2              | Class                 | 1:2    | 259568       |
| L3       | 2              | Class                 | 1:3    | 342448       |
| L4       | 2              | Class                 | 1:4    | 425328       |
| L5       | 2              | Class                 | 1:5    | 508209       |
| L6       | 2              | Class                 | 1:6    | 591089       |
| R1       | 2              | Species               | 1:1    | 82960        |
| R2       | 2              | Genus                 | 1:1    | 77293        |
| R3       | 2              | Genus                 | 1:1    | 93267        |
| R4       | 2              | Family                | 1:1    | 34457        |
| R5       | 2              | Family                | 1:1    | 40043        |
| R6       | 2              | Order                 | 1:1    | 70505        |
| R7       | 3              | Family and Order      | 1:1:8  | 290473       |
| R8       | 3              | Order and Phylum      | 1:1:8  | 374830       |
| R9       | 6              | Species, Order, Family and Kingdom | 1:1:1:2:1:4 | 588258    |

APPENDIX B ARCHITECTURE TYPES

In the training process for neural networks in phase 2 of MetaDEC, each architecture consists of four parts of encoder (θ), decoder (ϕ), critic (ψ), discriminator (ω). Each network is a multi layer perceptron. Each layer $i$ of a network is defined as follows.

$$x_i = g_i(W_ix_{i-1} + b_i)$$  \hspace{1cm} (11)

where $W_i$ and $b_i$ are layer’s parameters, $x_{i-1}$ is the output of previous layer, $g_i$ is the non-linear activation function. MetaDEC uses ReLU for all layers’ $g_i$, except the last layers of encoder, decoder and critic, which are set to be the identity function and the last layer of discriminator is set to be the sigmoid function. We initialized all layers’ parameters to random numbers drawn from a zero-mean Gaussian distribution with a standard deviation of 0.01. Tables 4, 5, 6, 7 describe the architectures used in our experiments for encoder, decoder, critic, and discriminator, respectively.

APPENDIX C VISUALIZATION OF LATENT SPACE

The visualization of latent spaces from results of cluster initialization (section II-A1 in Phase 2 of MetaDEC) for datasets S5 and R3 on four architectures, which are Tiny (Fig. 8), Small (Fig. 9), Large (Fig. 10), and Xlarge (Fig. 11).
FIGURE 8. Visualization of R3 and S5 on Tiny architecture.

FIGURE 10. Visualization of R3 and S5 on Large architecture.

FIGURE 9. Visualization of R3 and S5 on Small architecture.

FIGURE 11. Visualization of R3 and S5 on Xlarge architecture.