Metagenomic Binning using Connectivity-constrained Variational Autoencoders

Andre Lamurias [1 2] **Alessandro Tibo** [1] **Katja Hose** [1 3] **Mads Albertsen** [4] **Thomas Dyhre Nielsen** [1]

Abstract

Current state-of-the-art techniques for metagenomic binning only utilize local features for the individual DNA sequences (contigs), neglecting additional information such as the assembly graph, in which the contigs are connected according to overlapping reads, and gene markers identified in the contigs. In this paper, we propose the use of a Variational AutoEncoder (VAE) tailored to leverage auxiliary structural information about contig relations when learning contig representations for subsequent metagenomic binning. Our method, CCVAE, improves on previous work that used VAEs for learning latent representations of the individual contigs, by constraining these representations according to the connectivity information from the assembly graph. Additionally, we incorporate into the model additional information in the form of marker genes to better differentiate contigs from different genomes. Our experiments on both simulated and real-world datasets demonstrate that CCVAE outperforms current state-of-the-art techniques, thus providing a more effective method for metagenomic binning.

1. Introduction

Microbes influence all aspects of life, from our health to the production of enzymes and materials. Studying the genomes, functions, and interactions of microbial communities is beneficial to human life and our environment. As we only know the genomes of a small percentage of all existing microbes, more advanced methods are necessary for increasing the genome recovery rate. Metagenomics is a rapidly growing field focusing on the analysis and separation of DNA of entire communities of organisms. This is a complex problem since i) current high-throughput DNA sequencing technologies only produce fragmented genome reads and ii) due to the incompleteness of current reference databases, the full genome of most microbes in environmental samples remains unknown (Pasolli et al., 2019).

One of the key challenges in metagenomics is the ability to accurately group DNA subsequences according to their genomic origin, a process known as metagenomic binning. In general, binning is a two-step process, where the first step defines a notion of similarity between DNA sequences and the second step consists of grouping these sequences into clusters, which are referred to as bins.[1] The input to the binning process is a set of assembled contiguous DNA sequences (contigs). Contigs are obtained by representing the fragmented sequences as a graph, called an assembly graph, where each node represents a contig and the edges represent overlaps between contigs. Most binners only use local features of the individual contigs (Yang et al., 2021), thus failing to take full advantage of the relational information embedded within the assembly graph. Since, by construction, connected contigs share similar DNA sub-fragments, we hypothesize that the assembly graph holds potentially important information that can be exploited during the binning process.

The contribution of this work is a novel Connectivity-Constrained variational AutoEncoder (CCVAE) combining the advantages of existing binning approaches based only on local contig representations with domain-specific knowledge from the assembly graph and gene markers. Our model builds on top of the Variational Autoencoder (VAE) (Kingma & Welling, 2014) model proposed by Nissen et al. (2021), which we extend with domain-specific relational contig information. To the best of our knowledge, this is the first approach to leverage information from both the assembly graph and marker gene features within a probabilistic model for metagenomics binning. We report on extensive experiments using both simulated and real-world datasets for evaluating the performance of CCVAE and compare our approach to current state-of-the-art techniques. The results show that CCVAE outperforms current tech-

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark [2] NOVA LINCS, NOVA School of Science and Technology, Lisbon, Portugal [3] Institute of Logic and Computation, TU Wien, Vienna, Austria [4] Center for Microbial Communities, Aalborg University, Aalborg, Denmark. Correspondence to: Andre Lamurias <a.lamurias@fct.unl.pt>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1] In the remainder of this paper, the terms clusters and bins will be used interchangeably.

niques in terms of both accuracy and efficiency, with an improvement of 5.4% over the best existing method. The code and data used in the experiments are available at `https://github.com/MicrobialDarkMatter/ccvae`.

2. Domain background

The genome of an organism is the collection of all its genetic information, represented in the form of a sequence of DNA bases (`A T C G`). In an environmental sample, we encounter a combination of genomes from multiple individuals. The general metagenomic workflow starts by extracting and sequencing DNA fragments from the environmental sample. High-throughput sequencing devices produce raw electrical signals from the DNA fragments, which are converted into sequences over letters corresponding to the four DNA bases. This procedure generates millions to billions of such reads, which may originate from any of the genomes of all the organisms in the sample. A further complicating factor is that the reads only correspond to segments of the genomes, starting from random positions on the genome and with variable lengths, see Figure 1a. Depending on the technology used, reads are classified as short reads (100-150 bases) or long reads (2-30k bases). While longer read lengths are preferable for fully reconstructing the genome, up until recently long reads have also been more prone to errors (Sereika et al., 2021).

To obtain full microbial genomes, which are in the order of millions of bases, we need to combine the reads into longer sequences. As one environmental sample contains numerous identical copies of a microbial species, the reads will be a collection from these organisms starting at random locations on the genome and, hence, also have partial overlaps if enough reads are sampled. The process of combining these reads is called assembly and it involves finding overlaps between reads to obtain contiguous sequences, called contigs. Assemblers split reads into long k-mers and count how many times each k-mer occurs in the reads (see Figure 1b and c). The tool flye (Kolmogorov et al., 2020), for example, uses a k-mer size of 17 for generating the assembly graph. These k-mers are then structured in a de Brujin graph (Compeau et al., 2011) where the nodes are $(k-1)$-mers and they are linked by k-mers. Each edge adds another base to the sequence (see Figure 1d). Each walk of the de Brujin graph generates a sequence. Note that a node can have outdegree greater than one, leading to branches in the graph. For example, after the D edge in Figure 1 d, the path can continue through edge E or L, since both contain the $k-1$ sequence of that node (`A T T`). The de Brujin graph can be further simplified so that each node corresponds to the last letter of its $(k-1)$-mer and non-branching paths are merged (see Figure 1e). Finally, the nodes of the assembly graph (see Figure 1f) represent contiguous sequences of DNA bases (contigs), and the edges represent k-mers connecting contigs. The edges have a weight equal to the number of times the k-mer connecting the two nodes occurs in the data (see Figure 1c). Although this graph is directed, in practice we do not take into account the direction of the edges. This is because the sequence associated with the nodes is not used directly, and the features we extract from the contigs do not take into account the sequence.

Figure 1 shows an example assembly graph generation starting from the reads. In a real scenario, the reads are much longer, leading to longer contigs and, in practice, contigs smaller than 1kbp are discarded as these are assumed to be experimental artifacts. Furthermore, larger overlaps are also necessary to generate significant contigs, due to the repetitive nature of DNA. Assembly methods such as flye implement specific strategies to deal with repeats, which are sequences that repeat consecutively in genomes, leading to loops in the assembly graph. We extract features from the contig sequences, which we describe below.

Since the genome of each organism will be split into several contigs, advanced methods are required to recover high-quality genomes from a set of contigs. These methods are referred to as binners as they attempt to partition contigs into different genome-specific bins. As reads correspond to actual DNA sequences present in the sample, the read coverage of a contig will be correlated to the number of organisms in the sample. For example, if two organisms with the exact same genome exist in the sample, that genome will be sequenced twice, while a genome of an organism with double the number of copies in the sample will be sequenced four times. This property is called abundance and is a useful feature for binning contigs since contigs from the same genome should have similar abundance value (Albertsen et al., 2013). Read coverage is calculated by mapping the reads of the dataset to the contig sequences. Each base pair of a contig will overlap with zero or more contigs, therefore, the mean base coverage is used to represent the abundance of a contig.

Another useful property for binning are the k-mer frequencies of a contig, generally of size 3 or 4 (also known as k-mer composition). It has been shown that contigs from the same genome exhibit similar k-mer patterns (Burge et al., 1992). The k-mer frequencies are calculated by counting the number of times each of the possible k-mers occurs in the sequences. Note that this is different from the assembly process, where larger k-mer sizes are used. Longer k-mers would be less frequent and lead to a larger number of k-mer features, so smaller sizes are often more desirable for binning.

We can detect which genes are encoded in the contigs by comparing their sequences with reference databases when available. An important set of genes are the Single Copy

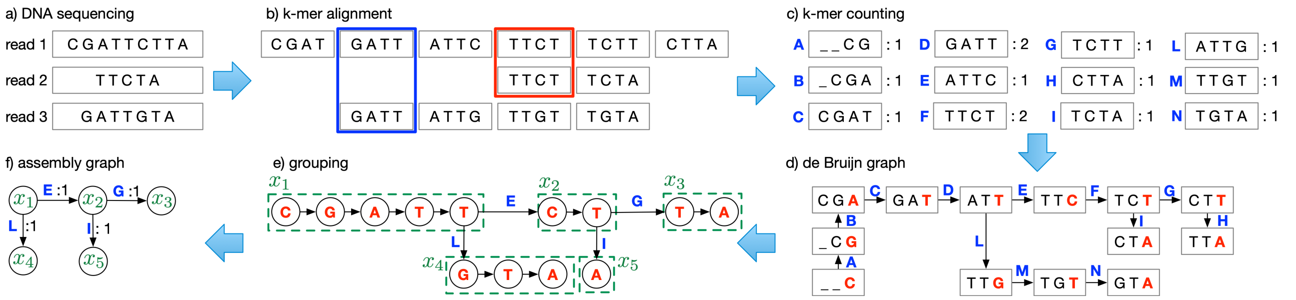

Figure 1. Assembly graph generation. a) The DNA sequences are read from an environmental sample. The raw signal is then converted into sequences of DNA bases (A G C T). b) While finding the best alignments, the reads are broken into k-mers ($k = 4$ in this example, but usually it is much larger), and matching k-mers are aligned. c) k-mers (denoted with bold blue letters for the sake of compactness) are associated with their multiplicity. d) k-mers are organized in a de Bruijn graph. Here, nodes are $(k-1)$-mers and edges are k-mers. e) Each sequential path is grouped into contigs x_1, x_2, x_3, x_4, and x_5. A contig contains a sequence of DNA bases (denoted with red letters in the de Bruijn graph). f) The assembly graph is finally produced, where nodes are contigs and edges are the multiplicities of k-mers connecting the contigs.

Genes (SCG), which occur only once in the full genome but which are essential for the functioning and reproduction of the microbes. Information about the SCGs can be incorporated into the binning process since two contigs with the same SCG must belong to different genomes and should therefore appear in different bins. From this perspective, an element of the binning task is to partition contigs into bins that contain a single copy of all the genes in the set of SCGs.

3. Related Work

In recent years, several binners have been proposed based on k-mer composition and abundance features (Yang et al., 2021). One of the best-performing binners based on these features is MetaBAT2 (Kang et al., 2019). MetaBAT2 uses these two features to compute a pairwise distance matrix for all contig pairs, calculated with a k-mer frequency distance probability and abundance distance probability. The former is based on an empirical posterior probability obtained from a set of reference genomes. A graph-based clustering algorithm is then used to bin the contigs based on their distances, where the contigs are linked according to their similarity scores. MaxBin2 (Wu et al., 2016) is another method that uses an Expectation-Maximization algorithm to estimate the probability of a contig belonging to a particular bin. The SCGs associated with each contig are used to estimate the number of bins. Although more k-mer composition and abundance methods have been proposed (Lu et al., 2017; Yu et al., 2018), MetaBAT2 and MaxBin2 are the most established and commonly used ones.

More recently, deep learning-based methods have been used to improve metagenomic binning. Deep learning models present an advantage over other statistical methods since the former types of models have the potential to learn more complex patterns in the data that would otherwise be difficult to model with standard methods. VAMB (Nissen et al., 2021) is a binner based on a Variational Autoencoder (Kingma & Welling, 2014) that encodes k-mer composition and abundance features in a low dimensional embedding, which is subsequently used for clustering/binning. However the usage of deep learning for metagenomics is still in its early stages and very few works have otherwise explored how to adapt existing algorithms for the metagenomics domain, in particular for data generated by more recent sequencing technologies that produce longer reads (Sereika et al., 2021).

Some recent works have attempted to use the assembly graph to improve metagenomic binning. The common assumption is that contigs that are linked in the assembly graph should also be binned together. For example, GraphBin (Mallawaarachchi et al., 2020) refines bins from other tools using information from the assembly graph. Specifically, GraphBin navigates the assembly graph using a label propagation algorithm and refines clusters that were separated in the binning process, but which nevertheless contain contigs that are linked in the assembly graph. Recent models use Graph Neural Networks (GNN) to learn features from the assembly graph. RepBin (Xue et al., 2021) proposed a Graph Convolution Network-based method that was tested on assembly graphs with high homophily. GraphMB (Lamurias et al., 2022) is also based on GNNs, but using a VAE for training contig-specific features which together with a GNN provided graph-level contig representations for clustering. Other recent works also incorporate some sort of additional information into the model specification or during model training (Pan et al., 2022; Kieft et al., 2022; Mallawaarachchi & Lin, 2022), but none of these capture the domain information contained in both the assembly graph and SCGs.

Table 1. Symbol definitions

SYMBOL	DEFINITION
x_t	k-mer frequencies
x_a	abundance features
l	length of node in base pairs
n_t, n_a	dimension of feature vectors
w_t, w_a, w_{kl}	k-mer, abundance and KL weights
$\hat{x}_t, \hat{x}_a$	reconstructed features
μ_z, σ_z	mean and variance of node representations
G	Assembly Graph
u, v	nodes in G
$w(u, v)$	weight of edge between u and v
z_l	contig-specific representations
z_g	graph representations
y	node labels
$\mathcal{G}_M$	set of single-copy genes
$\hat{\mathcal{Y}}$	single-copy genes of a node

4. Methodology

In the following, we use x to denote vectors in $\mathbb{R}^n$ (including scalars) and $\mathcal{X}$ for sets. In CCVAE, the data is always represented as an assembly graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. Each node $u \in \mathcal{V}$ correspond to a contig of length $\ell(u) \in \mathbb{N}$, represented as a tuple of features $x^u = (x_t^u \in \mathbb{R}^{n_t}, x_a^u \in \mathbb{R}^{n_a})$, where x_t^u are the contig's k-mer frequencies and x_a^u represents the relative abundances. In all experiments, we consider x^u as the concatenation of x_t^u and x_a^u which has size $n_t + n_a$. The dimensionalities n_t and n_a of both vectors depend on the specific datasets. Each node $u \in \mathcal{V}$ is either associated with a genome (categorical) label y^u or a set (possible $\emptyset$) of SCGs $\hat{\mathcal{Y}}(u)$ (up to 104) when genome labels are not available. The SCGs are inferred by CheckM (Parks et al., 2015), a standard metagenomic evaluation tool. Note that in both scenarios CCVAE remains completely unsupervised with respect to the genome labels, which are only used in the quantitative evaluations. The set of edges $(u, v) \in \mathcal{E}$ represents the pairs of nodes connected by the assembly graph. We adopt this more compact notation instead of a sparse adjacency matrix. See Table 1 for a summary of the symbols used throughout this paper.

The edges in the assembly graph do not necessarily imply that the nodes should have the same label, due to sequencing errors and genomes with similar sequences, creating erroneous edges in the assembly graph. To mitigate this issue, each edge $(u, v) \in \mathcal{E}$ is assigned a weight $w(u, v) \in [0, 1]$, which represents the normalized multiplicity of the k-mer that supports that edge (see Figure 1) and can thus be seen as the edge confidence. Here, 0 and 1 mean low and high normalized multiplicity, respectively. Nodes connected by edges with higher normalized multiplicity are more likely to belong to the same genome and are therefore more likely to have the same label.

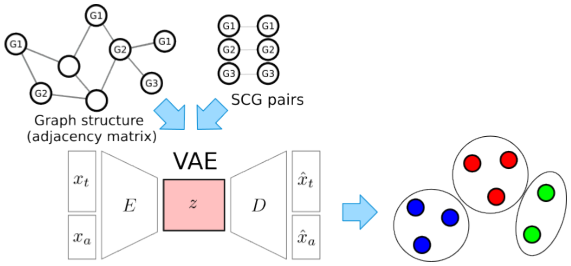

Figure 2. A Variational Autoencoder is used to learn node representations z_ℓ. The graph structure and node pairs with same SCGs are used to guide the training process so that contigs with same SCG (represented by G1, etc. inside each node) are pushed away and contigs connected by the graph are pushed together. The final representations are clustered into metagenomic bins and evaluated.

We learn contig representations by encoding k-mers x_t and relative abundances x_a with a VAE (see Figure 2). A VAE consists of an encoder E, parameterized by θ_E and a decoder D, parameterized by θ_D, where $\theta = \theta_E \cup \theta_D$. Each x_t is normalized to have zero mean and unitary variance, while each component of x_a is normalized to have a sum equal to 1 across all the relative abundances. If $n_a = 1$, we do not do this normalization but instead normalize x_a the same way as x_t.

The loss function used for training the model consists of three components

$$L = L_{vae} + L_e + L_{scg},$$

where L_{vae} captures the loss related to the contig-specific features and L_e and L_{scg} incorporate information about the assembly graph and SCGs, respectively.

4.1. Learning contig-specific representations

The loss function related to the contig specific features is adopted from (Nissen et al., 2021) and consists of three components[2]:

$$\begin{aligned} L_{vae}(x_t, x_a; \theta_E, \theta_D) =& L_t + L_a + L_{kl} \\ =& w_t \left\| x_t - \hat{x}_t \right\|^2 \\ & - w_a \, x_a^T \log(\hat{x}_a + c) \\ & - w_{kl} \, D_{KL}(\mathcal{N}(\mu_z, \sigma_z) \,\|\, \mathcal{N}(0, I)), \end{aligned} \tag{1}$$

where D_{KL} is Kullback-Leibler divergence, c is a small constant which we set to e^{-9}, $(\mu_z, \log \sigma^2)$ = $E((x_t, x_a); \theta_E)$, and $(\hat{x}_t, \hat{x}_a)$ = $D(z; \theta_E)$. Thus, the reconstruction error is separated into two terms capturing the k-mer compositions and abundances of the contigs, respectively. The k-mer

[2] In all of our experiments, $w_a = (1 - \alpha) \log(n_a + 1)^{-1}$, $w_t = \alpha / n_t$, and $w_{kl} = (n_z \beta)^{-1}$, where n_z is the dimension of μ_z, $\alpha = 0.15$ and $\beta = 200$. See also (Nissen et al., 2021).

reconstruction error, represented by the first line in Equation 1, corresponds to the Mean Squared Error (MSE) loss. The abundance reconstruction error, represented by the second line, corresponds to the Cross-Entropy loss since the abundance features can be modeled as probabilities, and a softmax function is applied to the reconstructed abundance features $\hat{x}_a$ before calculating the loss. However, if $n_a = 1$, we do not use the softmax function on $\hat{x}_a$ but instead use the MSE loss for the abundance features also, in which case the abundance loss function becomes:

$$L_a = w_a \left\| x_a - \hat{x}_a \right\|^2. \tag{2}$$

Learning is performed using gradient descent with one Monte Carlo sample z for the latent representation, for which the reparameterization trick is used:

$$z = \mu_z + \epsilon \cdot \sigma, \tag{3}$$

where $\epsilon \sim \mathcal{N}(0, I)$.

Finally, we use $z = \mu_z$ produced by E as node features in the following sections.

4.2. Edge and SCG losses

We expand on the VAE model by introducing additional domain-specific information in the form of the assembly graph edges:

$$\begin{aligned} L_e(z^u, z^v; \theta) = &-w(u,v) \log(\sigma(< z^u, z^v >)) \\ &- Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(1 - \sigma(< z^u, z^{v_n} >)), \end{aligned} \tag{4}$$

where θ are the model parameters, σ is the sigmoid function, $< \cdot, \cdot >$ denotes the scalar product, $w(u, v)$ is the edge weight between nodes u and v, P_n is a negative sampling distribution, and Q is the number of negative sampled. This way, we are constraining nodes with edges between them to have more similar features z than randomly sampled nodes. This is equivalent to the weighted binary cross-entropy, considering node pairs in the assembly graph as the positive label.

Similarly, we incorporate the SCG annotation by considering that contigs with overlapping SCGs should be in different clusters, and therefore have different representations. This is captured by the loss component

$$\begin{aligned} L_{scg} = - \sum_{u,v \in \mathcal{V}} & \mathbb{I}[|\hat{\mathcal{Y}}(u) \cap \hat{\mathcal{Y}}(v)| > 0] \\ & \log(1 - \sigma(< z^u, z^v >)), \end{aligned} \tag{5}$$

where $\mathbb{I}$ is the indicator function and $\hat{\mathcal{Y}}(u)$ and $\hat{\mathcal{Y}}(v)$ are the sets of SCGs for nodes u and v, respectively. This loss encourages different features for nodes with the same SCGs. We want nodes with the same SCGs to be in different clusters, to reduce the contamination of the final results.

Algorithm 1 Training CCVAE

```
Input: K-mer features $x_t$, abundance features $x_a$, assembly graph $G$ , and single copy genes $\hat{\mathcal{Y}}$ network parameters $\theta = \theta_E \cup \theta_D$, number of batches $N$, loss coefficients $\alpha_e$, $\alpha_{scg}$
Initialize encoder and decoder network parameters $\theta_E, \theta_D$
while true do
  for $i = 1, 2, \ldots, N$ do
    Sample latent variable $z_i \sim q_\theta(z_i|x_i)$
    Compute reconstruction losses $L_t(x_t, , D(z_i))$ and $L_a(x_a, , D(z_i))$ and KL divergence loss $L_{kl}(z_i)$ ($L_{vae}$) (Equation 1)
    Compute edge prediction loss $L_e(G, z_i)$ (Equation 4)
    Compute total batch loss $L = L_{vae} + \alpha_e L_e$
    Update encoder and decoder network parameters $\theta$ using Adam optimizer on $L$
  end for
  Compute SCG loss $\alpha_{scg} L_{scg}(\hat{\mathcal{Y}}, z)$ (Equation 5) for all pairs and update encoder parameters $\theta_E$
end while
```

Algorithm 1 shows how we combine different training objectives into our training loop. We divide the edge set $\mathcal{V}$ into N mini-batches. The features of the nodes corresponding to those edges are fed-forward to the model and used to calculate the reconstruction and KLD losses. At the end of each epoch, we optimize for the SCG objective, since only a small percentage of the nodes of each dataset are annotated with SCGs, and many batches would otherwise not include node pairs with SCGs.

4.3. Clustering

For the sake of consistency, we adopt the same clustering algorithm as in (Nissen et al., 2021), which is a modified version of the k-medoids algorithm that does not require an initial specification of the number of clusters. The clustering algorithm receives as input the concatenation of the contig-specific and the graph-specific representations, i.e., $z^u = (z^u_\ell, z^u_g)$, and consists of a three-step process: First, a seed medoid is found by picking a random z^u associated with a node and calculating the cosine distance to all other z^v. If any node has more neighbors than the current medoid within a small radius, that node is picked as the new medoid. The second step consists in determining the cluster radius. The distance from the chosen medoid to all other nodes is calculated, and the algorithm tries to find an optimal distance threshold that includes most of the nearby nodes but is small enough to exclude distant nodes; this should correspond to a local minimum in a histogram plot of the distances. The third step consists in removing the nodes within that threshold from the list of nodes to cluster and returning to step one until no more unclustered nodes are left. A more detailed description of the algorithm can be found in (Nissen et al., 2021).

4.4. Evaluation

To evaluate the quality of the bins (clusters), we adopt the completeness (see Equation 6) and the contamination (see Equation 7) criteria. Both criteria are domain-specific and indicate the quality of the clusters, according to the Minimum Information about a Metagenome-Assembled Genome (MIMAG) standard set by the Genomic Standards Consortium (Bowers et al., 2017). Completeness indicates whether the genome is suitable for a specific downstream analysis, while contamination indicates the fraction of the genome that might be contaminated with sequences from other genomes. These two metrics are required to submit a genome to public databases and to report it in publications. Using these criteria, we can classify a bin as a High Quality (HQ) bin if completeness > 0.9 and contamination < 0.05[3].

The recommended way of calculating these metrics is to use the list of SCGs as ground truth (recall that these genes are present exactly once in the genomes of nearly all bacteria). Some SCGs are collocated, meaning that they are in close proximity in the DNA sequence, and so their occurrences are not fully independent. For this reason, the ground truth is defined in terms of a set of sets of SCGs, $\mathcal{G}_M$, where each set of SCGs represents a group of collocated SCGs.

The completeness of a bin is given by:

$$\textsc{Comp}(\mathcal{G}_M, \hat{\mathcal{Y}}) = \frac{1}{|\mathcal{G}_M|} \sum_{\mathcal{G} \in \mathcal{G}_M} \frac{|\mathcal{G} \cap \hat{\mathcal{Y}}|}{|\mathcal{G}|}, \tag{6}$$

where $\hat{\mathcal{Y}}$ represents the multiset of SCGs associated with the nodes of a single bin. The completeness takes value 1 (the maximum) when all genes from $\mathcal{G}_M$ are identified in the bin. Completeness can be associated with the concept of recall since it measures the fraction of retrieved genes in the bins.

The contamination of a bin is defined as

$$\textsc{Cont}(\mathcal{G}_M, \hat{\mathcal{Y}}) = \frac{1}{|\mathcal{G}_M|} \sum_{\mathcal{G} \in \mathcal{G}_M} \frac{1}{|\mathcal{G}|} \big(\sum_{g \in \mathcal{G}} \big(\sum_{y \in \hat{\mathcal{Y}}} \mathbb{I}[g = y] \big) - 1 \big), \tag{7}$$

where $\mathbb{I}$ is the indicator function which is 1 if g is equal to y, and 0 otherwise. Here, we assume that if $g \notin \hat{\mathcal{Y}}$ the innermost summation in Equation 7 is 0. There is no maximum value of contamination since it will depend on the number of times an SCG is duplicated, i.e., a value of 1 means that on average all genes from $\mathcal{G}_M$ are duplicated once, and 2 means that all genes have two additional copies on average.

For simulated datasets, the genomes in the dataset are known. Therefore, it is possible to map the node sequences to those genomes and obtain the ground truth genome label y^u of each node. We followed the evaluation criteria for simulated datasets with ground truth labels as described in (Meyer et al., 2018): using the AMBER evaluation tool, we evaluate the precision and recall of each bin according to the labels of the nodes that constitute the cluster. If a bin contains all the nodes associated with one label, then that bin will have a recall of 1, and if it does not contain nodes of any other labels, it will have a precision of 1. In these metrics, we also take into account the length sequences associated with the nodes, because longer sequences will have a larger impact on recovering the genome sequence than shorter sequences.

Average precision (AP), average recall (AR), and F1 are thus defined as follows:

$$\text{AP} = \frac{1}{K} \sum_{k=1}^{K} \frac{TP_k}{TP_k + FP_k} \quad \text{AR} = \frac{1}{K} \sum_{k=1}^{K} \frac{TP_k}{TP_k + FN_k}$$

$$\text{F1} = \frac{2 \cdot \text{AP} \cdot \text{AR}}{\text{AP} + \text{AR}},$$

where K is the number of clusters and

$$TP_k = \sum_{u \in C_k} \ell(u) \mathbb{I}[y^k = y^u] \quad FP_k = \sum_{u \in C_k} \ell(u) \mathbb{I}[y^k \neq y^u]$$

$$FN_k = \sum_{u \notin C_k} \ell(u) \mathbb{I}[y^k = y^u].$$

Here, y^k is the label associated with the cluster C_k, calculated as the majority label of the node labels belonging to C_k. Similar to the previous criterion, we considered as HQ bins those with > 0.9 recall and > 0.95 precision.

5. Experiments

In this section, we present the experimental setup we used to evaluate our approach on simulated and real-world datasets, as well as the results obtained using the metrics previously introduced.

5.1. Data

We perform experiments on one simulated dataset and six Wastewater Treatment Plant (WWTP) datasets (Table 2). These are the same datasets used by Lamurias et al. (2022), where more details about data generation and processing can be found. The WWTP datasets come from a previous

[3] HQ bins are also required to have the 5S, 16S and 23S rRNA genes and 18 tRNA genes, however, we did not check for these properties in this work.

Table 2. Datasets used in the experiments. STRONG100 is a simulated dataset, while the others are real-world datasets. n_t is the dimension of the k-mer frequency features and n_a is the dimension of the abundance features.

DATASETS	# NODES	# EDGES	n_t	n_a
STRONG100	852	1,952	136	1
AALE	45,831	33,173	136	4
MARI	41,559	35,001	136	4
DAMH	38,578	34,186	136	4
HJOR	21,589	9,742	136	4
HADE	79,609	44,470	136	4
VIBY	27,109	13,235	136	4

study (Singleton et al., 2021), where we have access to four samples for each WWTP. Recall that the sample of each treatment plant is associated with a set of contigs, hence the abundance vector of each contig is of length four with one entry for each sample.

While the simulated dataset has ground truth labels, mapping each node to a specific genome, for the real-world datasets we do not have access to this information and we instead follow common practice and estimate the quality of the binning results in terms of the number of high and medium quality bins (see Section 4.4). Note that our results differ from Lamurias et al. (2022) because in that work CheckM was used, which attempts to find the most specific set of SCGs for each bin. In this work, we used only the standard set of Bacterial marker genes. The details of the graphs of each dataset are reported in Table 2.

5.2. Parameters

The input dimensions of each dataset are specified in Table 2. The n_t value is the same for all datasets as we used k-mers of size 4 and aggregated k-mers that were the same as their reverse complement. Both the encoder and decoder of the VAE consist of two hidden layers with 512 nodes and leaky ReLU activations. μ_z and $\log \sigma_z^2$ have sizes 32 for the simulated and 64 for the real-world datasets. The VAE is trained using gradient descent for 1000 epochs with a learning rate of $1e^{-3}$. We used mini-batches of 256 edges and sampled 10 negative pairs from a uniform distribution. The loss coefficients were set to $\alpha_e = 0.1$ and $\alpha_{scg} = 0.3$, determined empirically through grid search on the Aale dataset. We explore the effect of these coefficients, as well as different configurations of the VAE, in Appendix A.

5.3. Results

We compare the results of CCVAE with four competitors on the same datasets, using the default values specified in the corresponding papers. All the methods take as input the contig sequences and their abundances. We also compare with our optimized implementation of VAMB (Nissen et al., 2021), which we call VAMB*. This is a special case of our method where the edge and SCG losses are not taken into account ($\alpha_e = \alpha_{scg} = 0$).

Table 3. Results on the simulated dataset. AP and AR denote the average precision and recall over all bins. The F1 score is calculated by considering the harmonic mean of the precision and recall. VAMB* refers to the model trained with ($\alpha_e = \alpha_{scg} = 0$), VAE+E refers to the model trained with $L = L_{vae} + L_e$ and VAE+E+SCG to $L = L_{vae} + L_e + L_{scg}$.

MODEL	AP	AR	F1
METABAT2	0.907	0.513	0.655
VAMB	0.967	0.791	0.870
MAXBIN2	0.856	0.761	0.806
GRAPHBIN	0.854	0.530	0.654
GRAPHMB	0.969	0.765	0.855
VAMB*	0.894±0.021	**0.864±0.006**	0.878±0.009
VAE+E	**0.978±0.001**	0.812±0.009	0.887±0.005
VAE+E+S	0.973±0.001	0.843±0.015	**0.903±0.009**

Specifically, we have compared our approach with MetaBAT2 (Kang et al., 2019) and MaxBin2 (Wu et al., 2016), which are generally considered state-of-the-art binners (Yue et al., 2020; Vosloo et al., 2021). In this comparison, we have also included VAMB and GraphBin (Mallawaarachchi et al., 2020), with the former being included as it is the only published binner that uses deep learning methods, and the latter because it also takes the assembly graph as input. GraphBin runs on top of another binner, so it requires the output of another binner as input. We used MetaBAT2 as the input to GraphBin as it obtained the best results among the three other binners we considered. We present the results of the simulated and real-world datasets separately due to the different evaluation metrics being used. We evaluate each of the four binners as well as CCVAE with the three variations. To show the stability of CCVAE, we ran the experiments three times.

5.3.1. SIMULATED DATA

Table 3 shows the results obtained on the simulated dataset, where the metrics are calculated on the ground truth labels of the contigs using the AMBER evaluation tool (Meyer et al., 2018). These results indicate how the methods work in a scenario where the original genome of each contig is known. In this scenario, our method obtained better results than the other baselines. Without taking into consideration additional features, the VAMB* model obtained the highest recall, however, the highest F1-score was obtained using both the assembly graph and SCGs while training (VAE+E+SCG model). This was an improvement of approximately 3.8% compared to VAMB, the best-performing system we compared with.

5.3.2. Real-world data

As shown in Table 4, we can see that our method outperforms the other binners in terms of total HQ bins recovered. By training our model with additional connectivity information, we can consistently obtain more HQ bins than every other baseline method. In particular, we outperform both VAMB and MetaBAT2, both of which only rely on local contig features and thus fail to take advantage of the relational contig information embedded within the assembly graph. Our VAMB* model trained without additional information outperformed the best-performing system on these datasets, MetaBAT2, in terms of the total number of HQ bins across all datasets. When training with the proposed method, the total number of HQ bins is also higher.

The improvement is consistent across the datasets, demonstrating the impact of our method to the metagenomics community. We saw a significant improvement on the Aale, Mari, and Hade datasets and comparable results on the Damh, Hjor, and Viby datasets. We recovered an average of 9.9 more genomes with our method when compared to the VAMB* baseline, and 13.2 more genomes when compared to MetaBAT2, a commonly used binner. This represents an improvement of 3.8% and 5.2%, respectively. Comparing to the VAMB* baseline, on the Aale dataset we recovered 4.6.0 more genomes, while on the Mari dataset we recovered 1.8 more and 1.1 more on the Hade dataset Furthermore, our results have low variances in general, showing that it can consistently obtain the same number of genomes.

As we do not have the ground truth for these datasets, we do not know how many genomes should be recovered from each dataset. It is therefore possible that for the Damh, Hjor, and Viby datasets, the results obtained by our method and MetaBAT2 are already close to the maximum number of genomes that can be recovered from that data.

Effect of edge loss In both Table 3 and Table 4 we can see in the last three rows how augmenting our model with knowledge from the assembly graph and SCGs improves the overall results. We first show the VAMB* model that is only trained with the reconstruction and KLD losses. Adding the assembly graph edges to the loss improves the results, obtaining more HQ bins on average on most datasets, and a higher F1-score on the simulated dataset. We observed that in the simulated dataset, the homophily is not as high as in other datasets (Xue et al., 2021) (56% vs 89-97%), since it is based on long-read sequencing technology, similar to our real datasets. This means that some contigs from different genomes will be connected in the graph. The improvement is therefore not as linear as expected, since we are not accounting for erroneous edges.

Effect of SCG loss On both types of datasets, we also see improvements when using the SCG loss in addition to the edge loss, but the increase is not as pronounced as when adding the edge loss. We see at least two possible reasons for this: 1) comparably fewer nodes are associated with at least one SCG, so the majority of the embeddings are not affected by this loss component; 2) few nodes with the same SCGs are connected in the assembly graph, so the edge loss may already encourage an increasing distance between these disconnected pairs of nodes.

6. Conclusion

This paper reports on interdisciplinary research between data science and bioinformatics, addressing the problem of metagenomic binning of contiguous DNA fragments (contigs). This activity is key for understanding the diversity and function of microbial communities, which have a direct impact on both health and the environment. We have proposed CCVAE, a novel method for learning feature representations for contigs, combining a variational autoencoder with domain features in the form of marker genes identified in the contigs as well as an assembly graph in which the contigs are organized.

We have compared CCVAE with other state-of-the-art metagenomic binning methods on both simulated and real-world datasets. We observe that by exploiting the relational information in the assembly graph, we can significantly increase the number of high-quality genomes recovered during the subsequent binning process as compared to the state-of-the-art baseline methods. On the simulated dataset, we saw a considerable improvement in the F1-score over the other methods. By leveraging the SCGs features, we can obtain additional improvements on both types of datasets.

This work represents an initial step in the exploration of graph learning methods for metagenomic binning and we believe that there are several promising directions for further work. Using Graph Neural Networks (GNN) is a natural next step on which we have run preliminary experiments. Previously, Lamurias et al. (2022) proposed a 2-step model, where local features of a GNN model are defined by the embeddings of a VAE. We intend to combine our loss function with this GNN-based model, and training it end-to-end, so that both the VAE and GNN models are trained at the same time. Additionally, we plan to refine the clustering step in order to better take into account the distribution of SCGs over the different clusters. This will involve refining the loss function to promote high completeness and low contamination of the clusters. We expect that these challenges will have an impact on both the machine learning and the bioinformatics communities.

Table 4. Results on real-world datasets, in terms of the number of High-Quality bins. VAMB and our methods (VAE, VAE+E, VAE+E+SCG) correspond to the average and standard deviation of 5 runs. The highest score on each dataset is bolded. VAMB* refers to the model trained with ($\alpha_e = \alpha_{scg} = 0$), VAE+E refers to the model trained with $L = L_{vae} + L_e$ and VAE+E+SCG to $L = L_{vae} + L_e + L_{scg}$.

Dataset	Aale	Mari	Damh	Hjor	Hade	Viby	Total
MetaBAT2	53	41	50	28	51	**30**	253
VAMB	42±2	37.3±5.7	41.3±1.5	22±1	47.3±1.5	19±2	208.9
MaxBin2	20	20	21	14	21	20	116
GraphBin2	16	21	23	16	15	16	107
GraphMB	46	**48**	43	25	52	23	237
VAMB*	55.8±1.3	45.6±0.9	**49.8±1.1**	26.2±0.8	51.3±2.1	27.6±0.9	256.3
VAE+E	58.8±1.4	47.8±0.9	**49.8±1.4**	**28.4±0.9**	52.0±1.1	29.0±0.9	265.8
VAE+E+SCG	**60.4±0.4**	47.4±1	**49.8±0.8**	27.8±1.2	**52.4±1.9**	29.0±0.8	**266.2**

Acknowledgment

This work was partially funded by VILLUM FONDEN under grant agreement number 34299. Alessandro Tibo is now working at AstraZeneca AB R&D, Gothenburg.

References

Albertsen, M., Hugenholtz, P., Skarshewski, A., Nielsen, K. L., Tyson, G. W., and Nielsen, P. H. Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nature biotechnology*, 31(6):533–538, 2013.

Bowers, R. M., Kyrpides, N. C., Stepanauskas, R., Harmon-Smith, M., Doud, D., Reddy, T., Schulz, F., Jarett, J., Rivers, A. R., Eloe-Fadrosh, E. A., et al. Minimum information about a single amplified genome (MISAG) and a metagenome-assembled genome (MIMAG) of bacteria and archaea. *Nature biotechnology*, 35(8):725–731, 2017.

Burge, C., Campbell, A. M., and Karlin, S. Over-and under-representation of short oligonucleotides in DNA sequences. *Proceedings of the National Academy of Sciences*, 89(4):1358–1362, 1992.

Compeau, P. E., Pevzner, P. A., and Tesler, G. How to apply de Bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.

Kang, D. D., Li, F., Kirton, E., Thomas, A., Egan, R., An, H., and Wang, Z. MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ*, 7:e7359, 2019.

Kieft, K., Adams, A., Salamzade, R., Kalan, L., and Anantharaman, K. vRhyme enables binning of viral genomes from metagenomes. *Nucleic Acids Research*, 50(14): e83–e83, 05 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac341. URL https://doi.org/10.1093/nar/gkac341.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Kolmogorov, M., Bickhart, D. M., Behsaz, B., Gurevich, A., Rayko, M., Shin, S. B., Kuhn, K., Yuan, J., Polevikov, E., Smith, T. P., et al. metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, 17(11):1103–1110, 2020.

Lamurias, A., Sereika, M., Albertsen, M., Hose, K., and Nielsen, T. D. Metagenomic binning with assembly graph embeddings. *Bioinformatics*, 38(19):4481–4487, 08 2022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btac557. URL https://doi.org/10.1093/bioinformatics/btac557.

Lu, Y. Y., Chen, T., Fuhrman, J. A., and Sun, F. COCACOLA: binning metagenomic contigs using sequence composition, read coverage, co-alignment and paired-end read linkage. *Bioinformatics*, 33(6):791–798, 2017.

Mallawaarachchi, V. and Lin, Y. Metacoag: Binning metagenomic contigs via composition, coverage and assembly graphs. In *International Conference on Research in Computational Molecular Biology*, pp. 70–85. Springer, 2022.

Mallawaarachchi, V., Wickramarachchi, A., and Lin, Y. GraphBin: refined binning of metagenomic contigs using assembly graphs. *Bioinformatics*, 36(11):3307–3313, 2020.

Meyer, F., Hofmann, P., Belmann, P., Garrido-Oter, R., Fritz, A., Sczyrba, A., and McHardy, A. C. AMBER: assessment of metagenome binners. *GigaScience*, 7(6): giy069, 2018.

Nissen, J. N., Johansen, J., Allesøe, R. L., Sønderby, C. K., Armenteros, J. J. A., Grønbech, C. H., Jensen, L. J., Nielsen, H. B., Petersen, T. N., Winther, O., et al. Improved metagenome binning and assembly using deep variational autoencoders. *Nature biotechnology*, pp. 1–6, 2021.

Pan, S., Zhu, C., Zhao, X.-M., and Coelho, L. P. A deep siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments. *Nature communications*, 13(1):1–12, 2022.

Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P., and Tyson, G. W. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome research*, 25(7):1043–1055, 2015.

Pasolli, E., Asnicar, F., Manara, S., Zolfo, M., Karcher, N., Armanini, F., Beghini, F., Manghi, P., Tett, A., Ghensi, P., et al. Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. *Cell*, 176(3): 649–662, 2019.

Sereika, M., Kirkegaard, R. H., Karst, S. M., Michaelsen, T. Y., Sørensen, E. A., Wollenberg, R. D., and Albertsen, M. Oxford nanopore R10.4 long-read sequencing enables near-perfect bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. *bioRxiv*, 2021.

Singleton, C. M., Petriglieri, F., Kristensen, J. M., Kirkegaard, R. H., Michaelsen, T. Y., Andersen, M. H., Kondrotaite, Z., Karst, S. M., Dueholm, M. S., Nielsen, P. H., et al. Connecting structure to function with the recovery of over 1000 high-quality metagenome-assembled genomes from activated sludge using long-read sequencing. *Nature communications*, 12(1):1–13, 2021.

Vosloo, S., Huo, L., Anderson, C. L., Dai, Z., Sevillano, M., and Pinto, A. Evaluating de novo assembly and binning strategies for time series drinking water metagenomes. *Microbiology spectrum*, 9(3):e01434–21, 2021.

Wu, Y.-W., Simmons, B. A., and Singer, S. W. MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics*, 32(4): 605–607, 2016.

Xue, H., Mallawaarachchi, V., Zhang, Y., Rajan, V., and Lin, Y. RepBin: Constraint-based graph representation learning for metagenomic binning. *arXiv preprint arXiv:2112.11696*, 2021.

Yang, C., Chowdhury, D., Zhang, Z., Cheung, W. K., Lu, A., Bian, Z., and Zhang, L. A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data. *Computational and Structural Biotechnology Journal*, 2021.

Yu, G., Jiang, Y., Wang, J., Zhang, H., and Luo, H. BMC3C: binning metagenomic contigs using codon usage, sequence composition and read coverage. *Bioinformatics*, 34(24):4172–4179, 2018.

Yue, Y., Huang, H., Qi, Z., Dou, H.-M., Liu, X.-Y., Han, T.-F., Chen, Y., Song, X.-J., Zhang, Y.-H., and Tu, J. Evaluating metagenomics tools for genome binning with real metagenomic datasets and cami datasets. *BMC bioinformatics*, 21(1):1–15, 2020.

A. Hyperparameter optimization

We performed several experiments to find the optimal architecture for the VAE, as well as to find the optimal loss coefficients α_e and α_{scg}. All experiments were run on the Aale dataset, and the results were averaged over 3 runs. We picked the values that obtained the highest average number of HQ bins. In case of a draw, we picked the one with the shortest running time.

Table 5. Effect of the number of VAE layers (encoder and decoder) on the HQ bins and running time.

VAE LAYERS	HQ	RUNTIME
1	56.7±1.9	1:33:02
2	**57.0±0.8**	2:01:09
3	54.8±1.8	3:01:42

Table 6. Effect of the number of hidden units on the HQ bins and running time.

HIDDEN UNITS	HQ	RUNTIME
32	11.0±4.3	1:34:29
64	38.0±2.4	1:46:52
128	49.7±0.5	1:48:40
256	54.3±2.1	1:45:57
512	**57.7±0.9**	1:46:04
1024	**57.7±0.5**	1:50:34

Table 7. Effect of the latent dimension size on the HQ bins and running time.

DIMENSION SIZE	HQ	RUNTIME
16	54.7±0.9	2:02:25
32	58.3±1.2	2:01:04
64	**60.0±1.4**	2:08:19
128	59.3±1.2	2:14:48

Table 8. Effect of the loss coefficients α_e and α_{scg} on the HQ bins. For brevity, we omit the standard deviations and show only the averages over three runs.

α_e/α_{scg}	0	0.1	0.2	0.3	0.5	1
0	57.0	58.3	55.3	55.0	50.3	47.7
0.1	58.3	58.3	58.7	**59.7**	59.0	57.3
0.2	59.0	57.3	57.3	56.7	57.3	57.7
0.3	57.0	54.7	55.7	56.3	55.3	54.3
0.5	49.0	50.7	50.0	49.0	48.7	48.7
1	43.0	42.3	41.3	41.0	41.7	45