Summary

Wastewater Treatment Plants (WWTPs) clean influent wastewater. One step in the cleaning process is biological cleaning where a mass of microbes called activated sludge cleans the influent wastewater. The activated sludge consists of many different species of bacteria all impacting the balance and efficiency of the cleaning process. This process is complex and can be challenging to manage effectively in order to avoid decreased performance of the WWTP, which can cause pollution of effluent water. The complexity of the process is partially due to the multi-way interactions between bacteria, affecting each others growth. Increased knowledge about the bacteria and their interactions could help predict future development in abundance and enable a more precise management of the cleaning process.

The aim of this paper is to discover interaction groups among bacteria in WWTPs. We do this by using abundance data of bacterial species collected from activated sludge. The used abundance data is collected from danish WWTPs, specifically Aalborg West WWTP, and is sampled over a period of five years. This results in time series data of each species' abundance.

Interactions between bacteria are highly complex and difficult to analyze. Scientists study these bacteria a few at a time and in isolated environments, making it near impossible to describe a real complex environment. The complexity of the data thereby makes it difficult to infer the interactions between bacteria and as of now no ground truth exists.

In order to divide the species according to interaction groups we propose to simplify the interactions by only considering the relationship between two species at a time. We call this pairwise univariate time series data. We propose a machine learning model to cluster the pairwise univariate time series data. We furthermore suggest to divide each of these pairwise univariate time series into subsequences denoted as windows. This is done in order to evaluate if clustering using a finer granularity of data results in a better clustering. To reduce our dataset further, and to simplify our understanding of interaction groups, we limit the pairwise univariate time series to only being pairs with a highly negative or positive correlation.

Our model builds upon two models: DPSOM and LIME. DPSOM is a deep neural network using a probabilistic clustering with Self-Organizing Maps. Our contribution to this method is adapting it to handle input as windows of pairwise univariate time series.When clustering using more than one window per pair, we combine the soft cluster assignments from PSOM for all windows in a pair to get a soft assignment for the original pair. The pair is then assigned to the cluster with the highest probability.

Since evaluating our method with ground truth is not a possibility, we evaluate it using other metrics. We use the average silhouette score of a whole cluster for hyperparameter tuning, where the silhouette score describes the compactness of a cluster and how well it is separated from other clusters. A metric we use for clustering evaluation is Pearson correlation coefficient, which is used to calculate the pairwise correlations in a cluster and discuss whether the clusters are based on bacterial interactions. Here we find that our method does not cluster based on the interactions between the species of a pair. We also evaluate our method using a LSTM prediction model with the intuition that a good clustering result is a cluster consisting of similar objects, and that predicting on similar object leads to a more accurate prediction. This is indirectly evaluated using the domain specific loss function, Bray-Curtis Dissimilarity, as the primary evaluation metric. Here we find that clustering on windows might be meaningful, however more extensive testing is required to form a final conclusion. Lastly we visually inspect the LIME explanations to see if they provide any meaningful explanation. We find that the explanations are meaningful, however they might be a bit inconsistent. The explanations seem to suggest that our model produces clusters of high and low abundance instead of clusters of different bacterial interactions.

Discover Bacterial Interactions by Combining Clustering of Pairwise Univariate Time Series and Explainability

Alexander de Linde Agerskov, Christian Bro Sørensen, and Trine Juhl Holmager

Aalborg University, Department of Computer Science,
Selma Lagerlöfs Vej 300, DK-9220, Aalborg Øst, Denmark

June 11, 2021

Abstract

Knowing bacterial interactions in waste water treatment plants (WWTPs) can help manage the cleaning process, optimize it, avoid eutrophication and avoid pollution of effluent waters. In this paper we propose to find bacterial interactions in WWTPs by clustering pairwise univariate time series consisting of bacterial abundances sampled from activated sludge. We do this by modifying a deep clustering method called DPSOM to take pairs of bacteria as input. We then propose to split these pairs into subsequences called windows and performing the clustering on these windows. These cluster of windows are then used to produce clusters of the original full-length pairs. To help understand the clustering of the pairs we provide visual explanations, with the LIME framework, of which features in a pair contribute to that pair's clustering. As the dataset contains no ground truth in terms of interactions, we propose to evaluate our model using non-standard clustering metrics such as Pearson correlation coefficient and cluster-based prediction in addition to the LIME explanations.

1 Introduction

Effectively cleaning human waste water released from cities and industries is an ongoing challenge to reduce the impact of modern civilization on the environment [1]. To avoid eutrophication and pollution of receiving natural waters, large amounts of biomass and inorganic nutrients such as nitrogen- and phosphorous compounds must be removed before the water can be safely released into the environment [1]. For the past century, this has largely been done by engineered microbiological systems at sophisticated waste water treatment plants (WWTPs) through the activated sludge (AS) process, where complex microbial communities present in the waste water are continuously being recycled and mixed with the influent waste water through a series of different tanks [2, 3]. These microbial communities consist of thousands of different species, mainly bacterial, each with a unique metabolism and thus role in the treatment process [4, 5].

For optimal performance of the WWTP, it is crucial to maintain a balance of the presence, or absence, and abundance of the individual species in the community to avoid major problems [4]. One such problem is poor sludge settling properties caused by an increased abundance of filamentous bacteria, where a large portion of the sludge cannot be recycled due to excessive foaming, resulting in poor performance of the WWTP [6, 7]. The AS community structure is to some extent being controlled by manipulating the physical environment around the community by utilizing different tanks with for example aeration or mixing, blocking of sunlight, etc, but is otherwise largely shaped by stochastic processes such as complex inter-species interactions [4, 8, 9].

Practically no microorganism lives in isolation in nature, but instead form complex interaction webs with other species, and elucidating these interactions is critical for a general understanding of the community, how it is shaped and how it can be manipulated to optimize waste water treatment

[8]. Inferring these interactions is difficult and is usually done with time-consuming experiments in small, simplified lab-scale reactors [10], where only a few species are analyzed at once, limiting its scalability to describe a real complex environment more easily. Recent advances in machine learning and artificial intelligence models provide an explorative way to elucidate these complex interactions. The aim of this study is to apply machine learning to a detailed time series dataset [11] of abundance of the AS microbial community from a Danish WWTP in order to find bacterial interactions.

The available dataset does not contain any ground truth in terms of interactions, which means that the method for discovering interactions has to be unsupervised. Clustering is an unsupervised process used to find underlying structure of datasets and grouping the similar structures. With the idea of similar interactions having similar structure, it would be possible to separate bacterial interactions into different clusters. As this dataset is high-dimensional and complex, traditional clustering methods such as, e.g., k-means [12] and hierarchical clustering will not suffice as they generally have poor performance on such data due to the *curse of dimensionality*. More modern clustering methods are better at handling the complexity of the abundance data. These are however yet to be applied to this data.

To overcome the issue of clustering high-dimensional data, methods that perform dimensionality reduction and feature transformation have been used, as they create a representation that is easier to cluster on [13]. *Deep Neural Networks* (DNNs) in the form of *Autoencoders* (AEs), *Variational Autoencoders* (VAEs) and *Generative Adversarial Networks* (GANs) are methods providing this ability [14, 15]. DNNs combined with clustering methods have proven to greatly increase the clustering performance [16, 17]. A clustering method, which has been proven to work great in combination with a DNN is *Self-Organizing Map* (SOM) [18]. SOM is a neural network that is trained to produce a topological representation (called a map) of the input space, and a clustering of the input [19]. A model that combines the SOM with a deep architecture is *Deep Probabilistic SOM* (DPSOM) [18]. In terms of clustering performance this method outperforms other methods combining DNNs and clustering [18]. Some of these other methods are *DEC* [20], *IDEC* [21] and *SOM-VAE* [22]. Common for the above DNN methods is that they offer little to no explanation in terms of insight to the reasoning behind the clustering result.

As the abundance dataset does not contain any ground truth information we are not able to evaluate the model in the traditional way with metrics such as *clustering accuracy* and *NMI*. Since it is not possible to evaluate the model on the clustering result directly, another way is to inspect which features the model finds important for a prediction and assess whether these features are sensible. One method offering this explanation is *Local Interpretable Model-agnostic Explanations* (LIME) [23]. LIME is a framework that by tweaking the input for a model can tell the end user which features from the input are decisive for the clustering result and to which degree. Assessing whether there is a meaningful real world relationship between the features that the model focuses on and its prediction, requires expert knowledge about the problem domain and the data. If such a real world relationship exists, it can help build trust in the model and its predictions.

To address the above mentioned issues, we propose a model that combines a modified DPSOM architecture with a LIME architecture. Due to the complexity of multi-way bacterial interactions, we only consider pairwise interactions in this paper, and leave multi-way interactions as future work. As we aim to find interactions between bacteria we propose to organize species in pairs resulting in a dataset consisting of pairwise univariate time series. We further propose splitting these time series into subsequences called windows. We then perform clustering on these windows, and transform the resulting clusters of windows to clusters consisting of the original full-length pairs. We do this in an attempt to increase the clustering performance. We use a modified DPSOM architecture because the model in its original form is incompatible with pairs and windows of pairs as input. To tackle the issue of having no ground truth we use Pearson correlation coefficient in addition to visually inspected explanations from LIME. As we do not posses expert knowledge in the area of bacterial interactions, we evaluate the explanations by visually inspecting if the important features are consistent within each cluster.

In extension to this, we also assess the quality of the clusters by performing prediction on the individual time series in each cluster using our LSTM model from last semester [24]. We do this as we last semester discovered that performing prediction on similar multivariate time series tends to

increase the prediction performance.

With the goal of this project being to develop a method for discovering interactions between species without having a ground truth, our contributions are:

1. A method to perform clustering on pairwise univariate time series data of different granularity, while explaining which features of the data are contributing most to the clustering result.

2. An extension to the LIME explainability framework offering united visual explanations to inspect and validate the proposed clustering method.

3. A way to mitigate the lack of ground truth by using alternative metrics (Pearson correlation coefficient and cluster-based prediction) to evaluate the clustering results.

4. An empirical assessment of whether splitting time series into subsequences, clustering the subsequences and then transforming the subsequence-clusters to clusters of the original full-length time series, improves clustering performance.

2 Related Work

2.1 Deep Clustering

Recent work show that deep clustering, which combines clustering algorithms with deep neural networks (DNNs), greatly improves clustering performance [16, 17]. The underlying idea is that DNNs embed the data into a subspace holding intrinsic features of the data, which should be more suited for clustering. Several deep clustering methods exists, where the first well-known deep clustering method *DEC* [20] contributed to making the field popular [17]. DEC uses a stacked autoencoder (SAE) to learn a latent representation of the data and cluster the representations. While learning this representation a cluster assignment hardening (CAH) loss is used to make the model self-supervised, by creating a target distribution to optimize towards. Other methods build upon DEC, e.g., *IDEC* [21], which tries to better preserve the feature space by keeping the decoder of the SAE and using both the reconstruction and clustering loss during training, in order to make the features more representative and meaningful.

A different method, *SOM-VAE* [22], also uses an autoencoder to learn a latent representation, but uses a Self-Organizing Map (SOM) to cluster in the latent space. The SOM visualizes the relationship between clusters in a two-dimensional grid making it easier for the end user to interpret and visually reason on. *DPSOM* [18] extends SOM-VAE with probabilistic clustering and the CAH loss from DEC to further improve upon the clustering performance.

When the input to the methods mentioned above is time series, then a single input is either *multivariate* as data from several time series to a given timestep or *univariate* being a single time series. To the best of our knowledge, no existing method considers paired univariate time series as input.

When clustering on time series data, most methods cluster on either *whole time series*, *subsequences of time series* or *single timesteps* [25]. To the best of our knowledge, no methods cluster on subsequences of time series in order to use the result to reassemble the original time series into meaningful clusters of full-length time series. Neither is no one investigating if clustering at this finer granularity can improve clustering performance compared to the same time series data clustered at coarser granularity, specifically when used on paired univariate time series.

2.2 Explainability

While DNNs can be good at approximating complex functions, they offer limited insight into their internal reasoning [26]. Therefore DNNs act as *black boxes* providing no information on which features that are crucial for determining the output [26]. There have been several studies trying to mitigate this. The research mainly divides into two categories: *inherently interpretable models* and *post-hoc explanations* [27]. The preferred option is the former, but it requires the ability to build a model

which is adequately accurate for the setting [27]. Should this not be possible, the latter method can be used [27]. An example of an inherently interpretable model is the prediction model *N-BEATS* [28], which has dedicated layers for *trend* and *seasonality* in its DNN architecture. The prediction output of these layers can then be inspected individually, to see how trend and seasonality contribute to the final prediction.

While inherently interpretable models try to make the models themselves more explainable, post-hoc explanation methods treat the models as black boxes and try to come up with explanations in other ways [26, 27]. Examples of post-hoc explanation methods are *SHAP* [29] and *LIME* [23]. They work by perturbing the input and observing the effect on the output, to determine the importance of the features in the original input.

LIME can be seen as a subset of SHAP, where the main difference between them is that SHAP considers all possible perturbations while LIME only considers a limited amount. Additionally, SHAP provides both locally and globally interpretability, whereas LIME is a local-only method. Local meaning that the method handles a single input at a time and learns a linear approximation of the actual model in the neighborhood of that input. Despite the benefits of SHAP, it is computationally expensive and during an empirical study, the explanations produced by LIME was preferred over explanations produced by SHAP by end users [26]. Although LIME works with time series it cannot present a unified explanation for multiple inputs (in this case a pair of time series), but instead considers each time series separately.

3 Methodology

3.1 Problem Definition

We define a time series b of a specific species a as an ordered sequence of real-valued data $b_a = \{b_{a,t}\}_{t=1}^{T}$, where t is a timestep. Given a dataset of time series of M species $\{b_a\}_{a=1}^{M}$ of length T, where $b_a \in \mathbb{R}^T$ and each individual time series is paired with every other time series $x_i = \{b_a, b_c\}$, where $x_i \in \mathbb{R}^{2\times T}$ and $a \neq c$. This results in $N = \binom{M}{2} = \frac{M(M-1)}{2}$ pairs. The set of all pairwise univariate time series can be denoted as $X = \{x_i\}_{i=1}^{N}$.

The goal is then to partition the pairs X into a set of clusters $\{S_k\}_{k=1}^{K}$ and to find a set of explanations $\xi(X)$ which provides insight into the reasoning of the clustering method. The architecture can be seen in Figure 1.

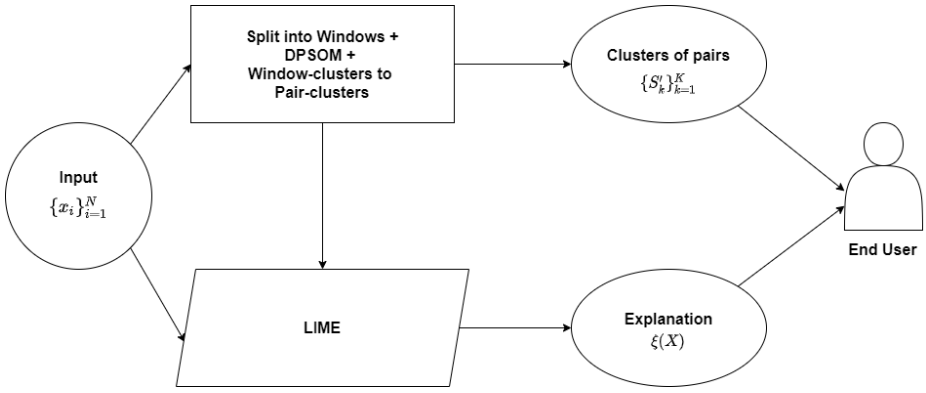


Figure 1: Method overview.

3.2 Split into Windows

To investigate whether clustering on subsequences of time series results in improved clustering performance, we split each full-length pair x_i into a set of U windows W_i. The set of windows is defined as $W_i = \{w_{i,j}\}_{j=1}^{U}$, where $w_{i,j} \in \mathbb{R}^{2\times V}$ and $V = \lfloor \frac{T}{U} \rfloor$. This can be seen in figure 2.

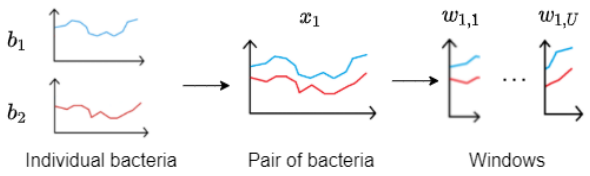


Figure 2: Process of arranging species in pairs and dividing them into windows.

3.3 Deep Probabilistic Clustering with Self-Organizing Maps

We follow the principles of Deep Probabilistic Clustering with Self-Organizing Maps (DPSOM) [18] for our clustering method. DPSOM is a clustering method consisting of a variational autoencoder (VAE) and a probabilistic self-organizing map (PSOM) using clustering hardening loss (CAH). We have tailored this method to our problem by adjusting the layers in the VAE, making it able to handle paired univariate time series as input. We have also altered the cluster assignment to output a probabilistic assignment to each cluster instead of an assignment to a single cluster. An overview of our version of DPSOM can be seen in Figure 3.

Variational Autoencoder

The VAE consists of an encoder f_ϕ and a decoder f_θ, which are deep neural networks. The input to DPSOM is windows of paired univariate time series $w_{i,j} \in \mathbb{R}^{2\times V}$ which are embedded into a latent representation $z_{i,j} \in \mathbb{R}^{V'}$ using the encoding part. The encoder f_ϕ consists of four layers. The first layer is a convolutional layer with a kernel of size 2×2 which is followed by three fully-connected layers. The purpose of the convolutional layer is to capture the relationship between the two univariate time series in a pair. The encoder maps the input $w_{i,j}$ to a latent embedding $z_{i,j}$ by learning a probability distribution $q_\phi(z_{i,j}|w_{i,j})$ which the latent embedding $z_{i,j}$ is sampled from.

The decoder consists of similar layers as the encoder but arranged in reverse order. The decoder reconstructs the original input $\hat{w}_{i,j}$ from the embedding $z_{i,j}$ by learning a probability distribution $p_\theta(w_{i,j}|z_{i,j})$ which the reconstructed instance $\hat{w}_{i,j}$ is sampled from. The loss contribution from the VAE is:

$$\mathcal{L}_{\text{VAE}} = \sum_{i=1}^{N}\Big[-\mathbb{E}_{q_\phi(z_{i,j}|w_{i,j})}(\log p_\theta(w_{i,j} \mid z_{i,j})) + D_{KL}(q_\phi(z_{i,j} \mid w_{i,j}) \parallel p(z_{i,j}))\Big] , \tag{1}$$

where the first term is the reconstruction loss which uses the expectation operator $\mathbb{E}$ to handle that we are sampling from a distribution. The second term encourages compactness in the latent space. Here $p(z_{i,j})$ is an isotropic Gaussian prior, with the purpose of ensuring that the learned distribution is close to the normally distributed Gaussian.

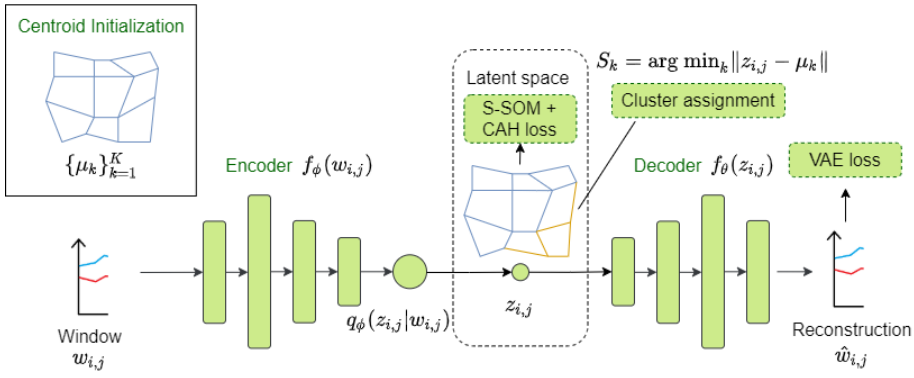


Figure 3: The architecture of DPSOM. A VAE maps a window $w_{i,j}$ to a latent embedding $z_{i,j}$. The embedding is clustered using PSOM, resulting in a probabilistic cluster assignment to each cluster.

Probabilistic Self-organizing Map

PSOM is a SOM with the CAH from DEC [20]. SOM is a clustering method, which organizes the cluster centroids in a topological neighborhood grid structure. It maps from a high-dimensional input space to a low-dimensional (often two-dimensional) *map space* which can be easily visualised. A SOM is composed of K nodes $S = \{S_k\}_{k=1}^{K}$, where each node S_k corresponds to a centroid $\mu_k \in \mathbb{R}^{V'}$ in the latent space of the VAE. For simplicity, we use a two-dimensional rectangular grid structure for the map. The size (number of rows and columns) of this grid is referred to as the SOM dimension (e.g. a SOM dimension of 2x2).

CAH is used to improve cluster purity which is achieved with the CAH-loss $\mathcal{L}_{\text{CAH}}$:

$$\mathcal{L}_{\text{CAH}} = KL\left(Y \| Q\right) = \sum_{i=1}^{N}\sum_{j=1}^{U}\sum_{k=1}^{K} y_{i,j,k} \log \frac{y_{i,j,k}}{q_{i,j,k}}, \tag{2}$$

where Q is Student's t-distribution with $q_{i,j,k}$ in (3) being the probability of assigning sample $z_{i,j}$ to cluster S_k. Y is the auxiliary target distribution, which is created from Q, but with extra emphasis on high confidence assignments. The clusters are then refined by iteratively forcing the distribution Q closer to the target distribution Y.

$$q_{i,j,k} = \frac{\left(1 + \left\|z_{i,j} - \mu_k\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{k'}\left(1 + \left\|z_{i,j} - \mu_{k'}\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}} \tag{3}$$

$$y_{i,j,k} = \frac{q_{i,j,k}^2 / \sum_{i'}\sum_{j'} q_{i',j',k}}{\sum_{k'} q_{i,j,k'}^2 / \sum_{i'}\sum_{j'} q_{i',j',k'}} \tag{4}$$

α in Student's t-distribution (3) is the degrees of freedom, and is set to 10 in DPSOM.

Before the training starts, the centroids μ_k are randomly initialized. During training an embedded sample $z_{i,j}$ is selected and the nearest centroid and its four immediate neighbors are updated to bring them closer to $z_{i,j}$. During the update, the SOM-like neighborhood structure is enforced with a *soft SOM loss*:

$$\mathcal{L}_{\text{S-SOM}} = -\frac{1}{N \cdot U}\sum_{i=1}^{N}\sum_{j=1}^{U}\sum_{k=1}^{K} q_{i,j,k} \sum_{h=1}^{H} \log q_{i,j,n_h(k)} \ , \tag{5}$$

where $n_h(k)$ returns the index of node S_k's h-th neighbor i.e. $n_2(1) = 3$ if node S_3 is node S_2's first neighbor in the SOM. The intuition is that if there is a high probability of $z_{i,j}$ belonging to cluster S_k, then the probability for $z_{i,j}$ belonging to the neighboring clusters $S_{n_h(k)}$ is also high.

Overall Loss

The overall loss used to guide the training of DPSOM is:

$$\mathcal{L}_{\text{DPSOM}} = \gamma\mathcal{L}_{\text{CAH}} + \beta\mathcal{L}_{\text{S-SOM}} + \theta\mathcal{L}_{\text{VAE}} \ , \tag{6}$$

where γ, β and θ are weights adjusting the reconstruction vs. clustering performance.

3.4 Window-clusters to Pair-clusters

For all N pairs where each pair have U windows, each window $\{w_{i,j}\}_{i,j=1}^{N,U}$ is now assigned to a cluster $\{S_k\}_{k=1}^{K}$ using DPSOM. The goal of this sub-process is then to transform the window-clusters $\{S_k\}_{k=1}^{K}$ to corresponding pair-clusters $\{S'_k\}_{k=1}^{K}$ consisting of the original full-length pairwise time series. For the sake of simplicity, we will only consider one pair x_i with its corresponding windows $W_i = \{w_{i,j}\}_{j=1}^{U}$ for the remainder of this section. As described in subsection 3.3, each window $\{w_{i,j}\}_{j=1}^{U}$ is assigned to the cluster they are closest to in the latent space. Instead of using this when

assigning pairs to clusters, we use the soft assignment scores $q_{i,j,k}$ of PSOM, which can be interpreted as how likely each window $w_{i,j}$ is to belong to each cluster S_k. We calculate the average probability of the soft assignments $q_{i,j,k}$ of all windows $\{w_{i,j}\}_{j=1}^{U}$ in a pair x_i to get a soft assignment for the complete pair (e.g. average of all windows probability of belonging to cluster one). The pair x_i is then assigned to the pair-cluster S'_k with the highest average probability of the soft assignments. This can be seen in Equation 7.

$$S'_k = \underset{k=1,\ldots,K}{\operatorname{argmax}} \frac{1}{U} \sum_{j=1}^{U} q_{i,j,k}, \tag{7}$$

3.5 Local Interpretable Model-agnostic Explanations

We follow the principles of the LIME framework [23] for our explanations. Our adaptation of the framework takes a pair as input and attempts to provide an explanation highlighting which sections of the pair influenced the prediction. The framework can be seen in figure Figure 4. The *original representation* of the explained pair is $x_i \in \mathbb{R}^{2\times T}$. We then look at the species of the pair in terms of T' slices of equal length. The activation of the slices can be denoted as $x'_i \in \{0,1\}^{2\times T'}$. This is the *interpretable representation*, which is a binary vector, where each slice corresponds to an entry in the vector. Initially each entry is given a value of 1 in the vector as all slices are activated at the beginning.

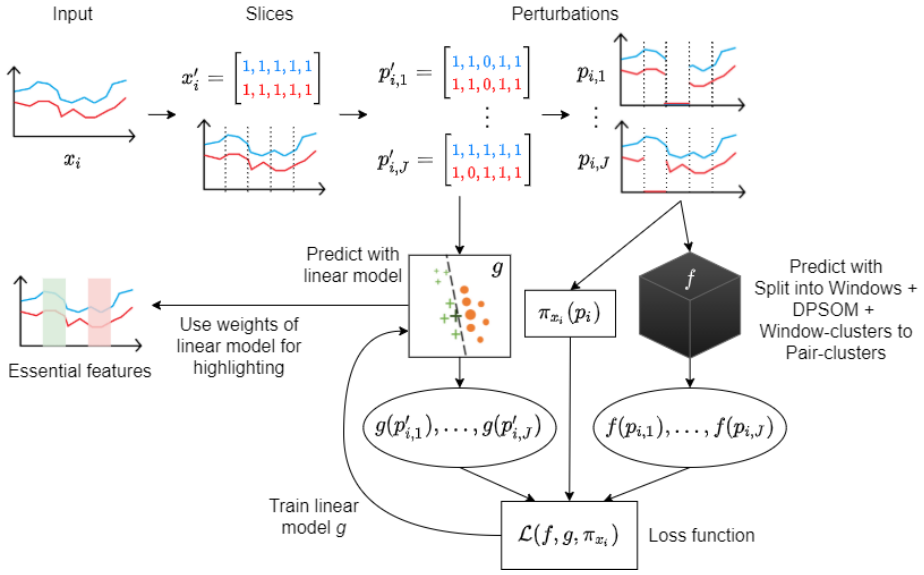


Figure 4: Framework for explaining the important features of a pair. The explained model f is treated as a black box. Linear model g is trained to approximate f using a loss weighted by the proximity $\pi_{x_i}(p_i)$ of a perturbed sample p_i to x_i.

An explanation is defined as $g \in G$, where G is the class of linear models. A linear model g can be presented visually to the user. When using linear models the explanation can be seen as a function $g : \{0,1\}^{2\times T'} \to \mathbb{R}$. Likewise the model being explained is a function $f : \mathbb{R}^{2\times T} \to \mathbb{R}$, i.e., $f(x_i)$ is the probability of x_i belonging to a specific cluster S'_k. Further, $\pi_{x_i}(p_i)$ is the distance between an instance of a pair x_i and a perturbed sample p_i. It is calculated using an exponential kernel with width σ: $\pi_{x_i}(p_i) = exp(-D_{cos}(x_i, p_i)^2/\sigma^2)$, where D_{cos} is the cosine similarity.

LIME calculates the following to produce an explanation:

$$\xi(x_i) = \underset{g\in G}{\operatorname{argmin}}\, \mathcal{L}(f, g, \pi_{x_i}) + \Omega(g) \tag{8}$$

$\Omega(g)$ is a measure of complexity of the explanation. This term is included to make the explanation easily interpretable. In our case of using linear models, $\Omega(g)$ is the number of weights ω_g in the linear model. $\mathcal{L}(f, g, \pi_{x_i})$ is a loss measuring how unfaithful g is to f by calculating the square loss between $f(p_i)$ and $g(p'_i)$ weighted by π_{x_i}:

$$\mathcal{L}(f, g, \pi_{x_i}) = \sum_{r=1}^{J} \pi_{x_i}(p_{i,r}) \left(f(p_{i,r}) - g(p'_{i,r})\right)^2 \tag{9}$$

The loss $\mathcal{L}(f, g, \pi_{x_i})$ is approximated by drawing J perturbed samples $p'_i \in \{0, 1\}^{2\times T'}$ in the neighborhood of x'_i weighted by π_{x_i}. This is done to learn the local behavior of f. A perturbed sample p'_i sampled in the neighborhood of x'_i has the same values as x'_i, except that some of the nonzero values are converted to zero. From the perturbed sampled p'_i in the interpretable representation we obtain the corresponding sample in the original representation $p_i \in \mathbb{R}^{2\times T}$. The sample p_i can then be fed to the model f to output cluster probabilities $f(p_i)$ which is compared to the probabilities given by $g(p'_i)$. The loss $\mathcal{L}(f, g, \pi_x)$ is then used to guide the training of the approximating linear model g from which we use the weights to highlight the B most influential slices. For each of these B slices we combine the weights of both species' contribution to the clustering result by adding them together, such that we have a single value per slice. This value denotes how much a specific slice contributes to the prediction that the pair belongs to the specific cluster. To limit the number of slices to B we define the measure of complexity as:

$$\Omega(g) = \infty \mathbf{1}\left[\|\omega_g\|_0 > B\right], \tag{10}$$

which implies that if the number of weights is greater than B, then the complexity is infinite, resulting in an infinite loss in (8).

4 Experiments

4.1 Data Description

We evaluate our model with the dataset called aalborg_west_species [11]. The dataset is a collection of bacterial abundances of species in Aalborg West WWTP over time. It consists of 1143 species and is sampled sequentially at 256 points in time from March 2015 to August 2020. The samples are however not sampled at equally distant points in time, but we do not take this into account for the experiments. As we pair each species with every other species, we would end up with far too many pairs if we used all species in the dataset. Instead we only use the 100 most abundant species, leaving us with 4950 pairs. As for the last filter criteria we omit the pairs where their Pearson correlation coefficient r satisfies $-0.5 < r < 0.5$. By only keeping the highly correlated pairs (positive and negative), we limit the range of possible interaction patterns in the data. We do this with the assumption that it makes it easier to evaluate the result using visual inspection. In total, 603 pairs, consisting of the 100 species, fulfill all criteria and will be used in the following experiments. To reduce noise we normalize each species abundance individually using division with maximum abundance. In extension to this, we smooth the species abundance using a moving average filter. We set the smoothing factor to 8.

4.2 Experimental Setup

DPSOM

We use the original DPSOM implementation [18], where we have modified it to work with pairs of time series as input. We modify the VAE to use a convolutional layer with kernel size 2x2 followed by fully connected layers of dimensions 500 - 500 - 2000 - latent dimension. To balance the contribution of the three different losses (in (6)), we set the multiplier θ to 0.05 after observing that the reconstruction loss dominated the overall loss during training when using higher values for θ. The choice of SOM dimension and the latent dimension is described in subsection 4.3. We train each model using 300

epochs and pre-train the VAE using 60 epochs. All other hyperparameters are as originally described in [18].

LIME

As the LIME [23] implementation does not support time series data we instead use an extension of the LIME framework that works with time series [30]. We have modified the explanations to output a single contribution per slice for the whole pair and not for each species individually. We split the pairs to be explained into $T' = 16$ slices, where we highlight the $B = 8$ (10) slices with the highest absolute weights. We draw $J = 10,000$ (9) perturbed samples for the pair to be explained.

Metrics

To measure the performance of the clustering and perform hyperparameter tuning we use five metrics: *Bray-Curtis Dissimilarity* [31], *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), *Pearson correlation coefficient* and *Silhouette score*. The first three metrics are loss functions used to evaluate the prediction of abundances - for each loss a lower value is preferred. Out of the three Bray-Curtis is the most important in our case because it is domain specific.

To perform the prediction we use our LSTM model from last semester [24]. The model works on species and not on pairs, therefore we need to separate the pairs into individual species in order to perform the prediction. The result from our method is clusters of pairs, which we need to transform to clusters of species. This is done by first counting the occurrences of each species in each of the pair-clusters and then assigning each species to the cluster in which it has the highest count. With clusters of species as input, the LSTM model can perform the prediction and report the average losses of each individual cluster. We denote the predicted abundance of species a as b'_a which along with the original abundance b_a can be used to calculate prediction losses with Bray-Curtis, MAE and MSE.

Bray-Curtis Dissimilarity is used since it is a metric often used in biology. We use it to quantify the dissimilarity of two samples and it outputs a value between 0 and 1. Here 0 means that the two samples are identical and 1 means that they are completely different. The formula for computing this metric is:

$$BC_{b_a b'_a} = 1 - \frac{2C_{b_a b'_a}}{AC_{b_a} + AC_{b'_a}}, \tag{11}$$

where $C_{b_a b'_a}$ is the sum of the smallest abundances in b_a and b'_a, AC_{b_a} is the sum of the original bacteria's abundance and $AC_{b'_a}$ is the sum of the bacteria's predicted abundance.

MAE is defined as:

$$MAE = \frac{1}{T}\sum_{t=1}^{T} |b_{a,t} - b'_{a,t}| \tag{12}$$

MSE is similarly defined as:

$$MSE = \frac{1}{T}\sum_{t=1}^{T} (b_{a,t} - b'_{a,t})^2 \tag{13}$$

Pearson correlation coefficient is used as a metric to measure the average correlation of the pairs in each cluster. We use this metric to see how the clustering reflect the pair-wise correlation. If the pairs are clustered on correlation, this can be seen as groups of different forms of interactions. Pearson correlation coefficient is calculated using the formula:

$$Pearson = \frac{\sum_{t=1}^{T}(b_{a,t} - m_{b_a})(b_{c,t} - m_{b_c})}{\sqrt{\sum_{t=1}^{T}(b_{a,t} - m_{b_a})^2 \sum_{t=1}^{T}(b_{c,t} - m_{b_c})^2}}, \tag{14}$$

where m_{b_a} is the mean of bacteria b_a abundance and m_{b_c} is the mean of bacteria b_c abundance.

Silhouette score [32] is used in hyperparameter tuning to choose the optimal number of clusters and latent dimension to cluster in. Here we use the Silhouette coefficients as a measure of how similar

a datapoint is in terms of its own cluster compared to how well separated it is from other clusters. This is done for all datapoints in one cluster to all datapoints in other clusters. The result is a score ranging from -1 to 1. A score near 1 indicates that a cluster is well separated from other clusters. A score near 0 indicates that clusters are overlapping. A score near -1 indicates low separation between clusters and that datapoints in the cluster are wrongly assigned. For each datapoint x_i in cluster S'_k $D_{intra}(x_i)$ is the mean distance from datapoint x_i to all other datapoints in that cluster and thereby a measure for how well x_i fits in the cluster. A small value is best.

$D_{inter}(x_i)$ is used to calculate the mean distance from a single datapoint x_i to all datapoints in another cluster different from the one x_i is assigned to. This value describes how separated x_i is from its closest neighbouring cluster, where a high value is preferred. Using $D_{inter}(x_i)$ and $D_{intra}(x_i)$, the silhouette score can be computed for one datapoint x_i:

$$Silhouette(x_i) = \frac{D_{inter}(x_i) - D_{intra}(x_i)}{\max\{D_{inter}(x_i), D_{inter}(x_i)\}} \tag{15}$$

The silhouette score can be calculated for all datapoints in a cluster. The average of these scores is computed to get the total score describing how compact the cluster is and how well it is separated from other clusters. In the silhouette method, the distance between two pairs x_1 and x_2 is calculated as:

$$D_{pair}(x_1, x_2) = \sqrt{\sum_{a=1}^{2}\sum_{t=1}^{T}(x_{1,a,t} - x_{2,a,t})^2}, \tag{16}$$

where $x_{1,a,t}$ is used to denote the abundance of species a of pair x_1 at timestep t. In (16) when $a = 1$ it refers to the first species in a pair (and $a = 2$ refers to the second species).

4.3 Hyperparameter Tuning

In order to produce better results we perform hyperparameter tuning on the SOM dimension and latent dimension. We perform the tuning with the pairs being split into four windows. In the tuning we trained 5 models for each hyperparameter combination of the chosen dimensions. We report the silhouette scores for the models with the best total loss (see (6)) in Table 1.

SOM Dimension	Latent Dimension 10	Latent Dimension 50	Latent Dimension 100
1x2	0.0741	**0.0745**	0.0710
2x2	0.0385	0.0421	0.0304
3x3	0.0018	0.0019	0.0118
4x4	-0.0054	-0.0015	-0.0199
5x5	-0.0423	-0.0223	-0.0284

Table 1: Silhouette scores for different latent dimension and SOM dimension combinations.

Number of Clusters

In order to deduce the optimal number of clusters representing interaction groups, we tested our model with five different SOM dimensions as can be seen in Table 1. The optimal number of cluster is chosen as the one with the highest Silhouette score, which in our case is 0.0745, which was achieved using a SOM dimension of 1x2. As we cannot completely trust the silhouette score, because it seems to be biased towards smaller SOM dimensions, we do not want to exclude all other SOM dimensions. We will therefore also perform the prediction experiments using a SOM dimension of 3x3.

Latent Dimension

The size of the latent dimension describes how many variables an input is represented by when encoded. If the size of it is too high, it can encapsulate non-intrinsic features, affecting the representation negatively due to representing unnecessary noise. If it on the other hand is too low, it fails to capture

the necessary features to describe the input which also leads to a poor representation. The size of the latent dimension highly affects the performance of the model since it is used for representing and clustering the input. To find an optimal size of the latent dimension we tested our model using three different sizes: 10, 50 and 100. The results can be seen in Table 1. We found that a latent dimension of 50 gave the best result in terms of silhouette score.

4.4 Results

Prediction

The prediction results for the modified DPSOM with the SOM dimension set to 1x2 can be seen in Table 2. The table also shows the result of prediction using two other clustering methods: *functionality* and *IDEC*, which originates from our project last semester [24]. To produce a fair comparison both of these method are also set to produce two clusters. The *functionality* method consists of assigning the species with the functionality *PAO* to one cluster and the species with the functionality *GAO* to another cluster (see [11] for further information on functionalities). The *IDEC* method is a deep clustering method which produces a clustering of the species.

With the intuition that clustering on windows of the pairs improves the performance of clustering, we tried splitting the pairs in different sizes. From Table 2 we see that the result of clustering on windows does not perform better in terms of prediction loss. On the contrary it performs slightly worse the more windows we use, however the difference is not significant. Based on these prediction results it does not seem that splitting the pairs into windows provides any benefits to the clustering.

When comparing the modified DPSOM method to last semester's methods, we observe that modified DPSOM (1 window) performs almost identical to IDEC. Together, IDEC and modified DPSOM (1 window) are the best performing clustering methods in terms of the prediction loss metric Bray-Curtis. We also observe that predicting on clusters representing bacterial functionalities outperform all other methods in terms of MSE and MAE.

LSTM prediction metrics using two clusters			
Clustering method	Bray-Curtis (avg)	MSE (avg)	MAE (avg)
Functionality	0.146	**0.095**	**0.241**
IDEC	0.132	0.141	0.268
Modified DPSOM (1 window)	**0.131**	0.140	0.248
Modified DPSOM (4 windows)	0.136	0.191	0.278
Modified DPSOM (8 windows)	0.143	0.190	0.289
Modified DPSOM (16 windows)	0.142	0.201	0.288

Table 2: Prediction loss metrics for each of the clustering methods using two clusters. For each clustering method the average of both clusters is used.

To investigate further whether this is only the case when using two clusters we perform the prediction with the SOM dimension set to 3x3. Similarly, we set IDEC to cluster into nine clusters. These prediction results can be seen in Table 3. We observe that modified DPSOM (1 window) and modified DPSOM (8 windows) are the best performing clustering methods in terms of the prediction loss metric Bray-Curtis. This shows that, as opposed to the results in Table 2, that clustering on windows might be meaningful. It can also be seen in Table 3 that all modified DPSOM methods heavily outperform IDEC.

LSTM prediction metrics using nine clusters			
Clustering method	Bray-Curtis (avg)	MSE (avg)	MAE (avg)
IDEC	0.166	0.161	0.247
Modified DPSOM (1 window)	**0.114**	**0.101**	0.222
Modified DPSOM (4 windows)	0.122	0.121	0.228
Modified DPSOM (8 windows)	**0.114**	0.114	**0.221**
Modified DPSOM (16 windows)	0.117	0.126	0.233

Table 3: Prediction loss metrics for each of the clustering methods using nine clusters. For each clustering method the weighted average of all clusters is used (weighted by the number of species in a cluster).

Something to keep in mind is that we do not perform the prediction on the actual results for the modified DPSOM. As we perform the prediction on the species instead of the pairs, the clustering loses some meaning, and the prediction results might therefore not accurately reflect the quality of the clustering of the pairs.

Correlation

As a measure of whether the clusters could represent different bacterial interactions, we use Pearson correlation coefficient measuring the correlation between the species of a pair. The idea is that the correlation of a pair can indicate certain types of interactions, e.g., a negatively correlated pair indicates that the species compete with each other while a positively correlated pair indicates beneficial cooperation. Therefore, if the goal is to group pairs into different interaction patterns, then the correlation of all pairs in one cluster should be different to the correlation of all pairs in another cluster. As seen in Figure 5, the pairwise correlations in each cluster are very similar. This indicates that our method does not succeed in grouping into distinctive interaction patterns.

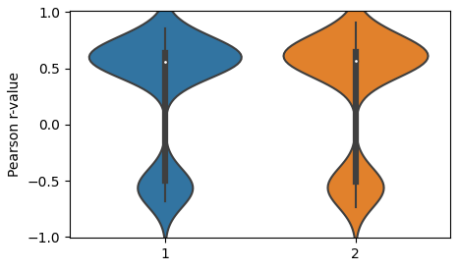


Figure 5: Violin plot of pairwise Pearson correlation in the two clusters found with modified DPSOM (4 windows).

4.5 Explanations

To explain which parts of the time series our model focus on, we use LIME to generate a set of explanations. We present the explanation for a model using $U = 4$ windows. For both clusters we randomly choose five pairs to explain (two from each cluster can be seen in Figure 6 and the last six are in Figure 7). Each explanation shows why the specific pair should belong (or not belong) to the cluster they are assigned to. In the LIME plots green slices should be interpreted as contributing to the pair belonging to the assigned cluster (here cluster 1 or cluster 2). Red slices are evidence against it belonging to the assigned cluster. The intensity of the color reflects the slice's contribution to the clustering, where a more intense color means a bigger contribution.

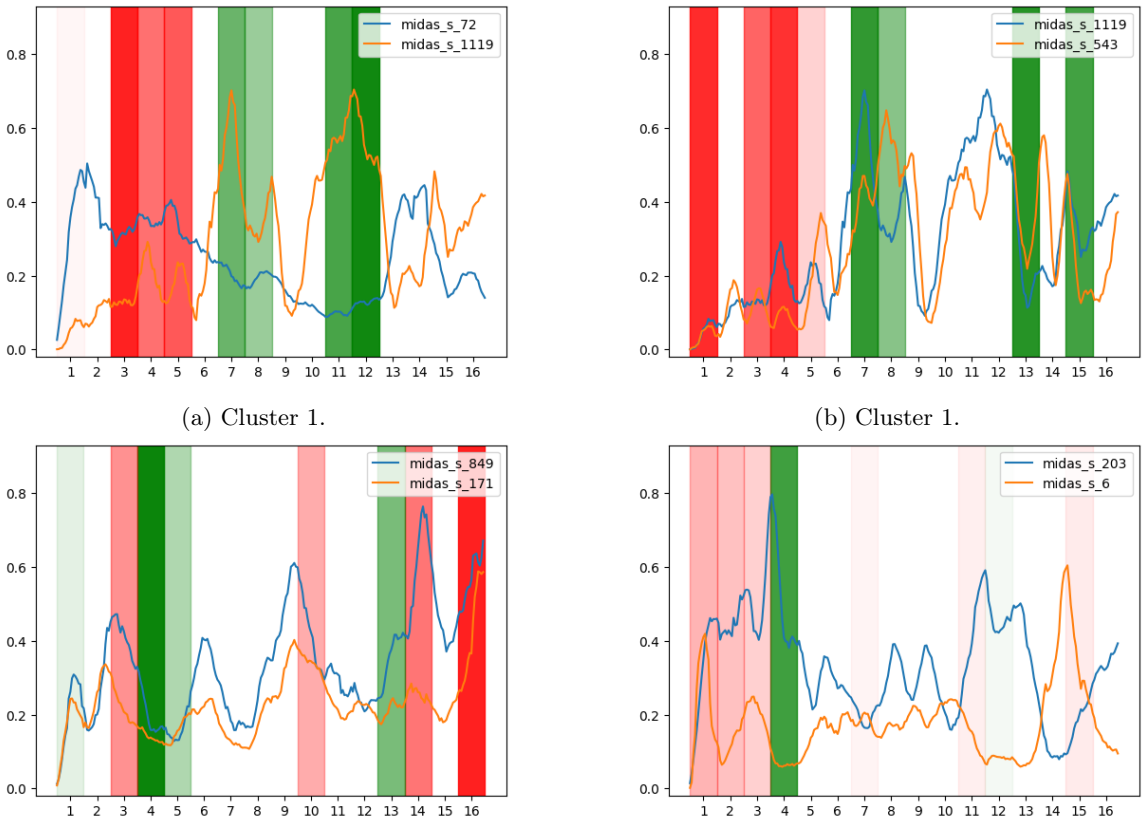
In Figure 6b slice 8 and 11 are fairly similar in terms of abundance and interaction, however only slice 8 is highlighted. The main difference between these slices is that `midas_s_543` (orange) has a higher abundance than `midas_s_1119` (blue) in slice 8 and a lower abundance in slice 11. This signals that it does matter which of the species has the highest abundance. When combining the time

series of two species to form a pair, they will have an order, i.e., one of the species is the first element of the pair while the other is the second element. This order is reflected in the latent encoding since the VAE is trained to reconstruct the input, and it therefore needs to consider the abundances of the species separately. We are however not concerned with the order of the species but instead the interactions between them, making this a flaw in the method.

When looking at the explanations in Figure 6a (e.g. slice 11 and 12) and Figure 6b (e.g. slice 7 and 8) it looks like DPSOM tends to cluster high abundance into cluster 1. For Figure 6c (e.g. slice 4 and 5) and Figure 6d it looks like they are clustered into cluster 2 on low abundance. Although this seems to be a pattern in terms of describing the clustering, it can also be seen that high or low abundance is not a stand alone factor deciding if it should be in a cluster or not, as there are also high and low abundances without any coloration (e.g. Figure 6d slice 7).

Another thing to notice is that the slices advocating for the current clustering of a pair does not seem to be conditioned on a single type of relationship between the species. An example of this can be seen in Figure 6b, where the species are negatively correlated in slice 8 but rather positively correlated in slice 13, which signals different types of relationships. Both slices are however interpreted by LIME as contributing to that the pair should belong to cluster 1.

From the above observations it is clear that no single factor is decisive for deciding which cluster a pair is assigned to. This can partly be explained by that LIME finds a local explanation and that the explained factors therefore only apply in that local neighborhood. However, most of the time the decisive factor seems to be whether the species' abundance is high or low, making the explanations consistent most of the time.



(a) Cluster 1.

(b) Cluster 1.

(c) Cluster 2.

(d) Cluster 2.

Figure 6: Four examples of LIME explanations. X-axis and Y-axis denote slice number and bacterial abundance respectively. The darker the color the more the slice is emphasized by the model.

5 Conclusion

We evaluated the model using prediction, correlation and visual explanations. The LIME explanations suggest that our method does not focus on the interaction between the species in a pair but instead focus on high and low abundance. This is supported by Figure 5 of the Pearson correlation of all pairs in each cluster, where we find that both clusters are very similar, indicating same interaction patterns in both clusters. The prediction suggests that it might be meaningful to split the pairs into windows, however with the lack of ground truth we deem it inconclusive.

Future Work

We believe the proposed method can perform better and have several suggestions in order to improve it. One suggestion is to change the way the latent encoding is done. For now the encoding captures which time series comes first in a pair, which distorts the clustering result since the order of the pair is not important when looking at the interaction. The encoding should instead have more focus on the relationship between the species. Another improvement to the method is to perform optimization on more hyperparameters, as we have only optimized some of them. In terms of evaluating the resulting cluster, our model might perform better if the prediction could be done on pairs instead of prediction using the unique occurring species for each cluster of pairs.

In order to make the method more useful it is relevant to scale the discovery of interactions to consider multi-way interactions and not only pairwise.

Acknowledgements

We would like to thank our project supervisor Chenjuan Guo for guidance throughout the project. We would also like to thank Kasper Skytte Andersen from Center for Microbial Communities, Aalborg University, for providing data and insight to the problem domain.

References

[1] C. L. Grady Jr, G. T. Daigger, N. G. Love, and C. D. Filipe, *Biological wastewater treatment*. CRC press, 2011.

[2] D. Orhon, "Evolution of the activated sludge process: the first 50 years," *Journal of Chemical Technology & Biotechnology*, vol. 90, no. 4, pp. 608–640, 2015.

[3] E. Ardern and W. T. Lockett, "Experiments on the oxidation of sewage without the aid of filters," *Journal of the Society of Chemical Industry*, vol. 33, no. 10, pp. 523–539, 1914.

[4] P. Gao, W. Xu, P. Sontag, X. Li, G. Xue, T. Liu, and W. Sun, "Correlating microbial community compositions with environmental factors in activated sludge from four full-scale municipal wastewater treatment plants in shanghai, china," *Appl. Microbiol. Biotechnol.*, vol. 100, pp. 4663–4673, May 2016.

[5] Y. Xia, X. Wen, B. Zhang, and Y. Yang, "Diversity and assembly patterns of activated sludge microbial communities: A review," *Biotechnology Advances*, vol. 36, no. 4, pp. 1038–1047, 2018.

[6] P. H. Nielsen, C. Kragelund, R. J. Seviour, and J. L. Nielsen, "Identity and ecophysiology of filamentous bacteria in activated sludge," *FEMS Microbiology Reviews*, vol. 33, pp. 969–998, 11 2009.

[7] D. Eikelboom, "Filamentous organisms observed in activated sludge," *Water Research*, vol. 9, no. 4, pp. 365–388, 1975.

[8] K. Faust and J. Raes, "Microbial interactions: from networks to models," *Nat. Rev. Microbiol.*, vol. 10, pp. 538–550, July 2012.

[9] F. Ju and T. Zhang, "Bacterial assembly and temporal dynamics in activated sludge of a full-scale municipal wastewater treatment plant," *ISME J.*, vol. 9, pp. 683–695, Mar. 2015.

[10] C. Li, K. M. K. Lim, K. R. Chng, and N. Nagarajan, "Predicting microbial interactions through computational approaches," *Methods*, vol. 102, pp. 12–19, June 2016.

[11] M. Nierychlo, K. S. Andersen, Y. Xu, N. Green, C. Jiang, M. Albertsen, M. S. Dueholm, and P. H. Nielsen, "Midas 3: An ecosystem-specific reference database, taxonomy and knowledge platform for activated sludge and anaerobic digesters reveals species-level microbiome composition of activated sludge," *Water Research*, vol. 182, p. 115955, 2020. Visited the associated webpage 03/06/2021.

[12] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (L. M. L. Cam and J. Neyman, eds.), vol. 1, pp. 281–297, University of California Press, 1967.

[13] I. Jolliffe, *Principal Component Analysis*, pp. 1094–1096. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[16] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," 2018.

[17] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.

[18] L. Manduchi, M. Hüser, M. Faltys, J. Vogt, G. Rätsch, and V. Fortuin, *T-DPSOM: An Interpretable Clustering Method for Unsupervised Learning of Patient Health States*, p. 236–245. New York, NY, USA: Association for Computing Machinery, 2021.

[19] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[20] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *CoRR*, vol. abs/1511.06335, 2015.

[21] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1753–1759, 2017.

[22] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Deep self-organization: Interpretable discrete representation learning on time series," *CoRR*, vol. abs/1806.02199, 2018.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016.

[24] A. de Linde Agerskov, C. B. Sørensen, and T. J. Holmager, "Predicting microbial communities in waste water treatment plants - exploring the effect of clustering when predicting multivariate time series," 9th semester project report, Dept. of Computer Science, Aalborg University, 2021.

[25] S. J. Zolhavarieh, S. Aghabozorgi, and T. Wah, "A review of subsequence time series clustering," *The Scientific World Journal*, 06 2014.

[26] J. V. Jeyakumar, J. Noor, Y.-H. Cheng, L. Garcia, and M. Srivastava, "How can i explain this to you? an empirical study of deep neural network explanation methods," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 4211–4222, Curran Associates, Inc., 2020.

[27] H. Lakkaraju, J. Adebayo, and S. Singh, "Explaining machine learning predictions: State-of-the-art, challenges, opportunities," in *Conference on Neural Information Processing Systems, NeurIPS-20*, 2020.

[28] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," 2020.

[29] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017.

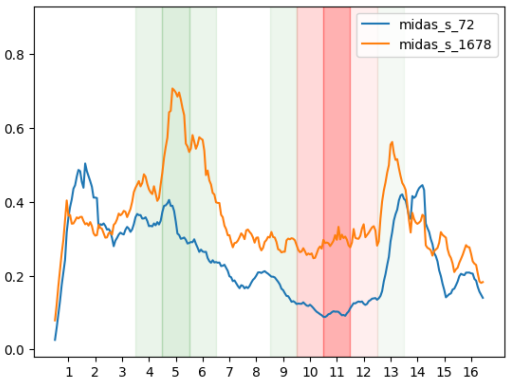
[30] M. T. Ribeiro, S. Singh, and C. Guestrin, "Lime for time," 2021. Last visited 25/05/2021.

[31] J. R. Bray and J. T. Curtis, "An ordination of the upland forest communities of southern wisconsin," *Ecological Monographs*, vol. 27, pp. 325–349, 1957.

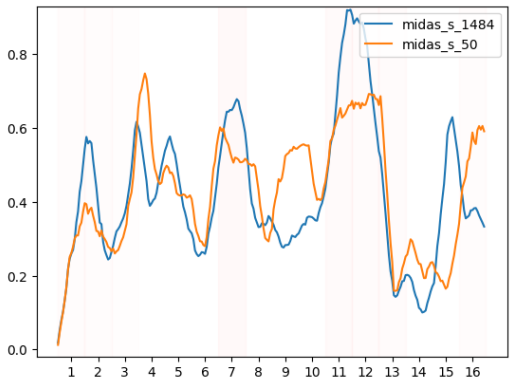
[32] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

Appendix

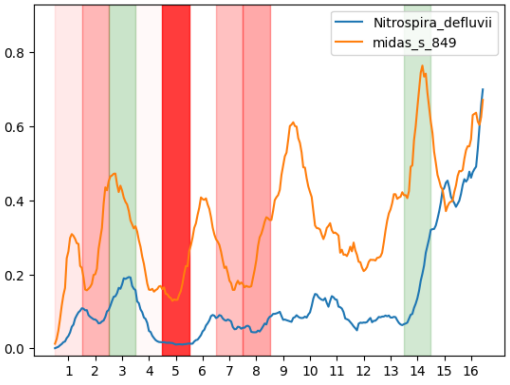
A LIME Explanations



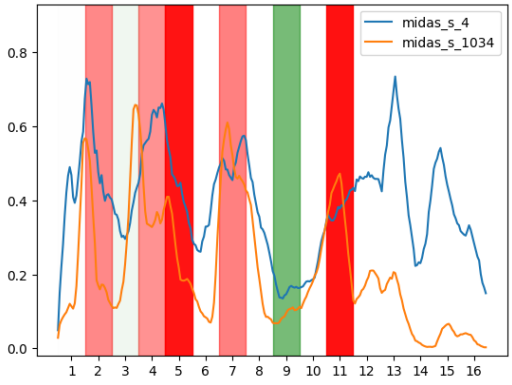
(a) Cluster 1.



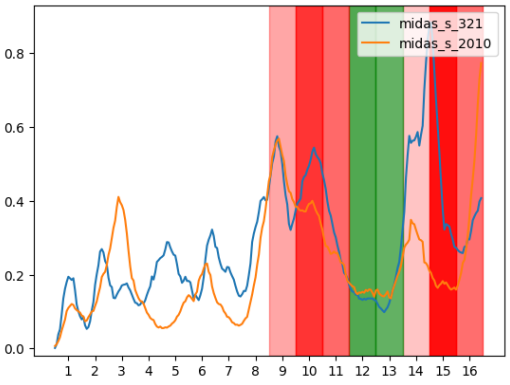
(b) Cluster 1.



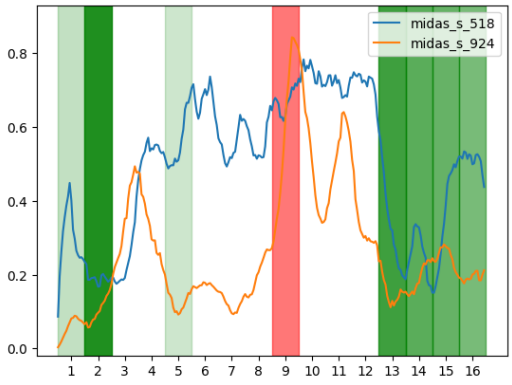
(c) Cluster 1.



(d) Cluster 2.



(e) Cluster 2.



(f) Cluster 2.

Figure 7: Six additional examples of LIME explanations. X-axis and Y-axis denote slice number and bacterial abundance respectively. The darker the color the more the slice is emphasized by the model.