PLOS ONE

RESEARCH ARTICLE

Identifying indicator species in ecological habitats using Deep Optimal Feature Learning

Yiting Tsai☯*, **Susan A. Baldwin**☯, **Bhushan Gopaluni**

Department of Chemical and Biological Engineering, University of British Columbia, Vancouver, Canada

☯ These authors contributed equally to this work.
* yttsai@chbe.ubc.ca

Abstract

Much of the current research on supervised modelling is focused on maximizing outcome prediction accuracy. However, in engineering disciplines, an arguably more important goal is that of feature extraction, the identification of relevant features associated with the various outcomes. For instance, in microbial communities, the identification of keystone species can often lead to improved prediction of future behavioral shifts. This paper proposes a novel feature extractor based on Deep Learning, which is largely agnostic to underlying assumptions regarding the training data. Starting from a collection of microbial species abundance counts, the Deep Learning model first trains itself to classify the selected distinct habitats. It then identifies indicator species associated with the habitats. The results are then compared and contrasted with those obtained by traditional statistical techniques. The indicator species are similar when compared at top taxonomic levels such as *Domain* and *Phylum*, despite visible differences in lower levels such as *Class* and *Order*. More importantly, when our estimated indicators are used to predict final habitat labels using simpler models (such as Support Vector Machines and traditional Artificial Neural Networks), the prediction accuracy is improved. Overall, this study serves as a preliminary step that bridges modern, black-box Machine Learning models with traditional, domain expertise-rich techniques.

Citation: Tsai Y, Baldwin SA, Gopaluni B (2021) Identifying indicator species in ecological habitats using Deep Optimal Feature Learning. PLoS ONE 16(9): e0256782. https://doi.org/10.1371/journal.pone.0256782

Editor: Zaher Mundher Yaseen, Ton Duc Thang University, VIET NAM

Received: May 3, 2021

Accepted: August 14, 2021

Published: September 10, 2021

Data Availability Statement: We have uploaded the pertinent raw data files, as well as the Python code containing data pre-processing steps, to the public repository GitHub (https://github.com/yitingtsai90/indicatorOTU).

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

The main motivation of this work is to propose a Machine Learning-based feature extractor which works generally for non-linear datasets of high dimensionality, then assess its efficacy on a real biological case with pre-determined features, to determine whether it is sufficiently reliable to be used for other similar studies. Biologically-related disciplines are often faced with the task of modelling on high-dimensional datasets (i.e. many raw input variables). In disciplines such as disease diagnosis and prevention, much of the current research focus is on maximizing prediction accuracy of outcome labels. For example, the authors of [1] compared and contrasted the performances across a large arsenal of predictive models (ex. Logistic Regression, Support Vector Machines, Random Forests, Naives Bayes *k*-Nearest Neighbors) for the task of diabetes-2 diagnosis. However, in other disciplines such as ecology, dimensionality reduction and feature engineering are arguably equally important tasks, which aid the intuitive understanding of these datasets. In ecological datasets, community interactions between individual species (the input variables) can be difficult to model. Therefore, tasks such as the identification of *indicator species* (i.e. keystone or leader species) can be challenging, as the user is often perplexed as to which dimensionality reduction algorithm(s) are suitable. In past literature, methods such as *IndVal* [2] have traditionally been used for indicator species identification. Successful implementations can be seen in publications such as [3–5]), where abundance counts of microbial species are available from various sites. The *IndVal* method uses Multi-Dimensional Scaling [6] followed by Correspondence Analysis (CA) and Detrended Correspondence Analysis (DCA) [7] to rank the topology of sites. The indicator species groups are then identified using using Hierarchical [8] and k-means clustering [9]. The final indicator value of a particular species can be roughly described as its frequency of occurrence across all clusters, across all sites. However, due to the ever-increasing complexity and non-linear nature of modern datasets, the use of these traditional statistical tools may reduce the overall algorithm's ability to generalize to a wide range of interactions between features.

Indicator species identification can be considered a special case of a more general category of modelling tasks, known as *feature analysis*. Given a data matrix $\boldsymbol{X}$ with N total samples as rows and d columns as raw variables, the precise definition of *feature analysis* can be subtly distinguished as the follows:

1. **Feature selection**: Determining whether each raw variable $x_j (j \in [1, \cdots, d])$ contributes significantly to the model output(s).
2. **Feature extraction**: Determining linear or non-linear combinations of raw variables (ex. $x_1 \cdot log(x_2)$) which contribute sigificantly to the model output(s).

The limitations of current state-of-art feature extraction methods are two-fold. The first problem is that many extractors simply use *feature selection* techniques, which are univariate in nature and are therefore unable to detect any multivariate interactions. More specifically, in some cases individual variables may not be relevant by themselves, but contribute significantly to the final outcome when they co-exist (in linear or non-linear fashions). The second problem is the limiting assumptions of the dimensionality reduction algorithms used in multivariate feature extractors. For example, extractors which prioritize the intuitiveness and interpretability of latent features often use Principal Component Analysis (PCA) [10] as a tool. This algorithm assumes that the underlying latent space is a linear combination of raw inputs (i.e. species), which may not be an accurate representation of many biological systems. Another example algorithm is k-means, which assumes that the underlying distributions of data are multivariate Gaussians (due to k-means being a special case of Gaussian mixtures [11]). If k-means were used to analyze a dataset which is highly non-Gaussian, then the results obtained could be misleading or meaningless.

In the current literature, most *feature selection* techniques are univariate approaches which assign "yes" or "no" labels to each raw variable, with respect to its relevance. A simple example is the addition of a ℓ_1 or *lasso* regularizer [12] to any model objective function. Other methods include *Mean Decrease in Accuracy (MDA)* and *Mean Decrease in Gini (MDG)* [13], which are used in conjunction with Random Forest (RF) models. In MDA, individual features are permutated (scrambled) randomly and the resulting effect on model accuracy is determined, compared to the base-case where no features are scrambled. In MDG, the feature importance is determined by observing how many times a feature is used to create a decision split in the RF model. The disadvantages of these methods are mainly their univariate nature, as well as significant false discovery (positive or negative) rates.

On the other hand, *feature extraction* techniques attempt to recognize multivariate interactions between raw variables. A detailed review of the state-of-art methods in this area can be found in the work of [14]. One straightforward and computationally-inexpensive method is the aforementioned PCA, which discovers linear combinations of raw inputs. A previous publication of similar topic is [15], which used PCA to determine proteomic biomarkers. Another successful example is the work of [16], in which the authors used an enhanced PCA algorithm to correctly determine the latent features which best predict diabetic retinopathy in patients. In other cases, however, PCA may yield poor approximations due to the presence of non-linear latent features. These non-linearities can be partially mitigated by the use of methods such as Isometric Mapping (ISOMAP) [17], Locally Linear Embedding (LLE) [18], *t*-Stochastic Neighbor Embedding (*t*-SNE) [19], and Uniform Manifold Approximation and Projection [20]. The aforementioned algorithms provide lower-dimensional representations of data which are easy to visualize. For instance, a recent paper by [21] showed that *t*-SNE can be used to group similar eco-provinces in order to elucidate the community structures in each region. However, a major shortcoming of these non-linear methods is that extracted latent features are difficult to express in terms of the original raw variables. An example of this is the Supervised Locally Linear Embedding (SLLE) paper by [22]. In this work, the authors showed that three flower classes within the *Fisher Iris* dataset [23] can be compressed down to three distinct regions in a 2-***d*** space. However, the intuitive relationships between the 2 latent components and the original 4 features are never explicitly explored. Most multivariate dimensionality reduction methods prioritize either understanding of extracted features (ex. PCA), or easy visualization of clusters (ex. *t*-SNE), but few exist that satisfy both. Finally, the choice of dimensionality reduction method also depends on the amount of domain knowledge available. For example, when the functional structure of the latent features is well-known, then *kernels* [24] (a list of basis functions such as polynomials, square roots, exponentials, etc.) can be used to pre-determine the desired feature representation. On the other hand, if little to no domain knowledge is available, then the user resorts to fitting each kernel to find an appropriate one. If the non-linear structure is too complex, then none of the pre-established basis functions are suitable. Therefore, there exists a clear need for a dimensionality reduction procedure, which does not rely on copious amounts of prior knowledge.

Recent advances in computing power have supported the emergence of a powerful class of models known as Deep Learning (DL) [25]. These models are capable of training highly-accurate models to solve *classification* problems, due to deep neural networks being good *universal approximators* [26, 27] (i.e. capable of fitting any non-linearity using activation functions, if trained sufficiently). Works such as [28–31] demonstrate the high predictive capabilities of DL used in supervised learning cases. In these studies, the desired features are either pre-engineered using PCA, or by neural networks in *black-box* fashions that render them difficult to interpret. In applications that prioritize predictive accuracy, knowledge of the exact identities of the latent features may not be important. However, in other applications that prioritize diagnosis and root-cause analysis, the feature extraction step becomes critical in telling the user which combination variables to seek, and which to avoid. To the best of our knowledge, our work is one of few which explores the efficacy of DL models when designed to address both fronts: predictive accuracy and ability to identify the correct associated features.

In this work, we focus the application of this DL model to bioinformatics problems, an example being ecological habitat prediction and identification of indicator species associated with each habitat. These datasets include species abundance counts as samples; each sample consists of a large number of taxonomic units identified by 16*S* *r*RNA amplicon sequencing, represented either as Operational Taxonomic Units (OTUs) [32] or Amplicon Sequence Variants (ASVs) [33]. The corresponding outcome is usually a class label distinguishing its source (ex. lake, river, ocean, etc.) or its habitat (natural, disturbed, etc.). The goal of predictive modelling is to estimate the correct habitat label of new samples, given their abundance counts of all species. While the model's test-set prediction accuracy is the focus of many ML research efforts, one can argue that remediation applications obtain more actionable information from feature extraction. Therefore, the motivation of our work in the context of bioinformatics applications can be clarified as follows:

1. Estimate the habitat label of each site using Machine Learning models, to produce a second set of labels which can be compared and contrasted to those obtained using traditional methods.
2. Identify the indicator species associated with each habitat, thereby eliminating the need to know abundance counts from all available species. This significantly reduces time and financial costs associated with manual sample collection and laboratory analysis.

We end this section by summarizing the scope, objectives, and advantages of our work as follows: develop a Deep Learning-based feature extractor for binary classification problems, with two main advantages over existing extractors:

1. It is agnostic to the true non-linear structure of the underlying feature space.
2. It still produces interpretable latent features.

We accomplish this by taking advantage of the universally-approximating property of Deep Learning, i.e. the ability of neural networks to learn any non-linear structure through brute-force combinations of activation functions (ex. *ReLU*, *SELU*, *sigmoid*, *tanh*, etc.). In order to maintain interpretability of the latent features, we use *ReLU* activations, which results in affine combinations of raw inputs, similar to those obtained from PCA. Finally, the general and case-specific objectives of our work can be summarized as follows:

1. Build and train a Deep Learning model for binary classification, which predicts the class labels of new samples with sufficiently high accuracy.
2. Identifies a latent space that optimally separates the two classes, which is then used to extract the most relevant features for prediction.
3. Demonstrate the efficacy of the proposed method on a real ecological dataset, by comparing the predicted class labels and extracted features with those identified previously using traditional methods.

In this paper, we will conform to standard Machine Learning notation regarding numerical matrices and vectors associated with datasets. The nomenclature will mirror those used in texts such as [25] and [11].

Nomenclature

The following is a compiled list of abbreviations used within this manuscript:

ANN: Artificial Neural Net
ASV: Amplicon Sequence Variants
CA: Correspondence Analysis
DCA: Detrended Correspondence Analysis
DL: Deep Learning
GAN: Generative Adversarial Network
IID: Independent and Identically Distributed
k: Number of latent features

$k^{[l]}$: Number of neurons in the l^{th} hidden layer of a DL model
ISOMAP: Isometric Mapping
LLE: Locally Linear Embedding
MA: Moving Average
MDA: Mean Decrease in Accuracy
MDG: Mean Decrease in Gini
MDS: Multi-Dimensional Scaling
ML: Machine Learning
NaN: Not a Number
OTUs: Operational Taxonomic Unit
PCA: Principal Component Analysis
PCoA: Principal Coordinate Analysis (equivalent to MDS)
*r*RNA: *r*-Ribosomal Ribonucleic Acid
RF: Random Forest
SVM: Support Vector Machine
t-SNE: *t*-Stochastic Neighbor Embedding
VAE: Variational Autoencoder
$\boldsymbol{x}^{(i)}$: The i^{th} data sample (all features)
$\boldsymbol{x}_j$: The j^{th} data feature (all samples)
$\boldsymbol{x}_j^{(i)}$: The i^{th} sample of the j^{th} data feature
$\boldsymbol{z}^{[l]}$: The l^{th} hidden layer of a DL model
$\boldsymbol{z}_j^{[l]}$: The j^{th} neuron in the l^{th} hidden layer of a DL model

Materials and methods

The proposed modelling and feature extraction workflow will be performed on datasets of a general nature, by adhering to the workflow depicted in the following Fig 1:

The paper will provide a brief introduction of the methods employed for Steps 1 and 2. Most of the focus will be emphasized on Steps 3 and 4, which describe the concept behind the novel Deep Learning feature extractor as well as the interpretation of its results.

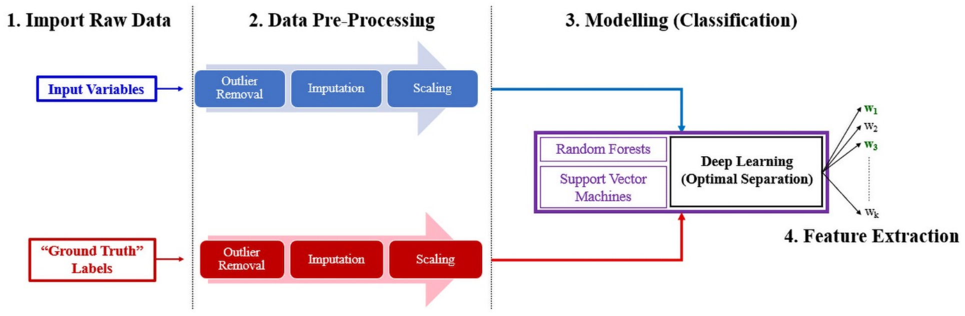


Fig 1. Overall data analysis workflow in block diagram form. *(Step 1)*: The collection of raw input data samples, as well as a corresponding set of labelled "ground-truth" targets. *(Step 2)*: The pre-processing of raw input data into suitable structures for modelling, guided by any available domain or expertise knowledge. *(Step 3)*: The training of several types of classification models (including Deep Learning), which maps inputs to their corresponding discrete class labels. *(Step 4)*: The design of special objective function within a Deep Learning classification, which identifies a latent space with improved class separation. The most dominant latent features are then distinguished by the magnitude (ex. ℓ_2 norm) of the neural network weights.

https://doi.org/10.1371/journal.pone.0256782.g001

Collection of raw data and ground truths

Any modelling task requires the availability of data regarding the system of interest. An example of a combined chemical-biological dataset is shown in the following Fig 2.

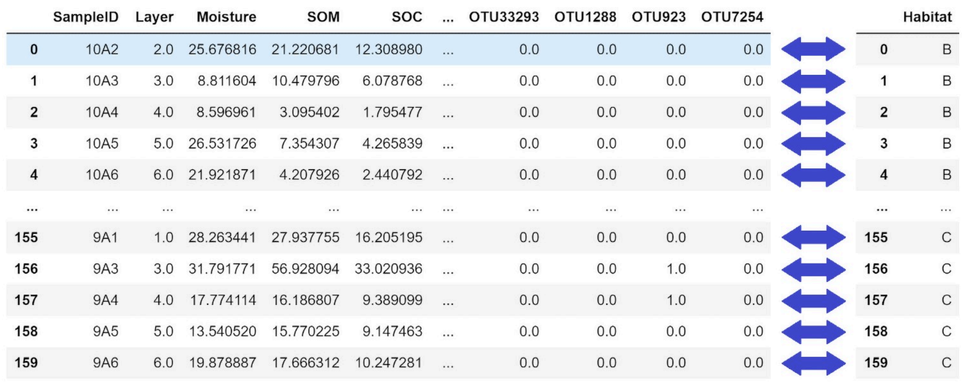
In accordance with ML literature, the raw input data are structured in a $\mathbb{R}^{N \times d}$ matrix that has N total rows representing the number data samples, and d total columns representing the number of raw variables. Inputs are usually continuous values; in some cases categorical labels may appear, but these can be discretized and transformed into numerical labels.

On the other hand, since this is a *classification* problem, the output or target variable is a set of discrete values. These represent the membership of each sample to a specific class, within a finite set of classes. Class labels can be easily transformed into numeric labels (ex. 0, 1, 2, etc.), even if they were not originally labelled as such. The accuracy of these outputs will inevitably affect the quality of any model trained using them. Therefore, the target variables for a dataset are often screened and scrutinized by domain experts and data scientists, before they can be ascertained as *ground truths*. They must be absolutely "correct" in the sense that few deviations from reality exist due to noise or other sources of error.

This work investigates the Mount Polley tailings storage breach, which occurred in the Quesnel region of British Columbia, Canada, in 2014. The paper [34] is one major publication which reports the disturbance to the soil and sediment microbiological communities (surrounding Mount Polley). The report is also accompanied by a dataset containing abundance counts of microbial species in every site; therefore, we will treat the findings and conclusions as "ground truths." In other words, the binary habitat labels of each collected sample—*natural* or *disturbed*—were determined by soil ecology experts and are thus considered baselines for any new classification models. The following section provides more details on this case study as well as its implications.

Ground truth case study: Mount Polley tailings storage breach. In 2014, the Quesnel region (located in central British Columbia, Canada) suffered a major ecological disturbance, caused by the breach of the Mount Polley tailings dam. The released tailings material travelled and deposited into the nearby Hazeltine Creek, Quesnel, and Polley Lakes. The physical, biological, and ecological impacts of this breach have been thoroughly explored in publications such as [35] and [34], in attempt to characterize both the spatial and temporal changes within the microbial communities.

This paper will focus and expand upon the established results in [34], which has attempted to model the microbial community shifts using 16*S r*RNA and full metagenome sequencing.



	SampleID	Layer	Moisture	SOM	SOC	...	OTU33293	OTU1288	OTU923	OTU7254
0	10A2	2.0	25.676816	21.220681	12.308980	...	0.0	0.0	0.0	0.0
1	10A3	3.0	8.811604	10.479796	6.078768	...	0.0	0.0	0.0	0.0
2	10A4	4.0	8.596961	3.095402	1.795477	...	0.0	0.0	0.0	0.0
3	10A5	5.0	26.531726	7.354307	4.265839	...	0.0	0.0	0.0	0.0
4	10A6	6.0	21.921871	4.207926	2.440792	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
155	9A1	1.0	28.263441	27.937755	16.205195	...	0.0	0.0	0.0	0.0
156	9A3	3.0	31.791771	56.928094	33.020936	...	0.0	0.0	1.0	0.0
157	9A4	4.0	17.774114	16.186807	9.389099	...	0.0	0.0	1.0	0.0
158	9A5	5.0	13.540520	15.770225	9.147463	...	0.0	0.0	0.0	0.0
159	9A6	6.0	19.878887	17.666312	10.247281	...	0.0	0.0	0.0	0.0

	Habitat
0	B
1	B
2	B
3	B
4	B
...	...
155	C
156	C
157	C
158	C
159	C

Fig 2. Printout of *pandas* dataframes containing raw data collected directly from ecological sites. *(Left)* Dataframe containing raw input variables. *(Right)* Dataframe containing output class labels.

https://doi.org/10.1371/journal.pone.0256782.g002

The main finding of [34] was that the identification of indicator species associated with natural and disturbed sites allowed the prediction of future biogeochemical trajectories in said sites. Specifically, ecologists could model just as accurately using the much smaller sub-set of bio-markers as when using the entire list of available species. This results in a significant reduction in both the complexity and cost of ongoing ecosystem health monitoring. Therefore, a strong justification can be made for the discovery of indicator species using other methods, such as Machine Learning, in order to complement those identified using the available statistical tools.

The data collected from the Mount Polley breach can be collected in a single input matrix $\boldsymbol{X}$, which contains $N = 70$ total samples as rows and $d \cong 22000$ total raw variables as columns. Out of the 70 total samples, 41 belong to an undisturbed habitat, while the remaining 29 belong to a disturbed habitat. The exact descriptions of these $C = 2$ habitats (or classes) can be expanded as follows:

1. **Class 0**: Natural habitat of organic-rich soil related to a wetland community.
2. **Class 1**: Disturbed habitat originally consisting of native subtrate, which has been replaced by a mixture of fine tailings and sands.

The exact analysis steps of this study can be summarized as follows:

1. **Step 1**: Label each sample as either *natural* (Class 0) or *disturbed* (Class 1).
2. **Step 2**: Identify indicator species associated with either habitat.

The proposed feature extractor in this paper will be used to find an optimal latent space $\boldsymbol{Z}$, which serves as a transient between the mapping from $\boldsymbol{X}$ to the target habitats $\boldsymbol{y}$. Note that the raw input matrix $\boldsymbol{X}$ is considered "small-N-big-d" (i.e. few samples compared to a large number of raw variables), with a significant amount of noise confounding the raw variable space. Thus, traditional classification methods (ex. SVMs, RFs, feedforward networks, etc.) and clustering methods (ex. PCA, MDS, etc.) are expected to produce confounded class predictions or clusters with significant overlap. The desired transformation of the data space is illustrated in the following Fig 3.

Pre-processing of biological variables

Due to the presence of microbial data in the form of species abundance counts, extra transformations were necessary to refine these entries. The abundance counts of many species are zero across most sites, thus resulting in a heavily low-skewing distribution. If these sparse counts are then used for modelling with no transformation, the models would place heavy emphasis on the many sparse species. This could result in spurious results caused by the vast negligence of highly-abundant but rarely-occurring species. In order to provide a more sensible count matrix which can be used for microbial community analysis, the following steps were taken:

1. **Removal of extremely sparse species**: The original count table includes 21721 total species. The abundance counts of each species was summed across all available samples. The counts of the top 50 species with respect to overall sum were selected as raw input variables.
2. **Log-transforms of remaining abundance counts**: In order to mitigate the low-leaning skew even more, a base-10 logarithmic transformation was applied to the remaining species counts, using the equation:

$$\boldsymbol{x}_{species,log} = log_{10}(\boldsymbol{x}_{species} + 1) \tag{1}$$

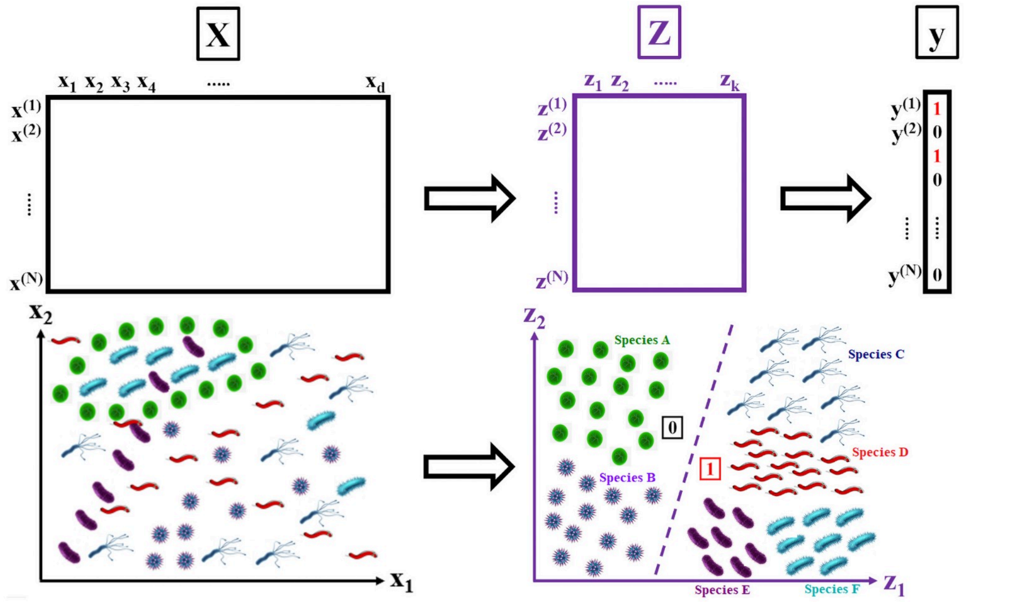


Fig 3. Discovery of an optimally-separating latent feature space. *(Top Left)* The high-dimensional and confounded raw inputs ***X*** makes class separation a challenging task.*(Bottom Left)* A case example including six microorganism species (A through F) which are entangled in the raw input space. *(Top Right)* A DL model which learns a latent space ***Z*** that optimally separates the classes. *(Bottom Right)* The disentanglement of the six species into distinct classes, which can be further aggregated into two major classes—Class 0 (A and B) and Class 1 (C through F).

https://doi.org/10.1371/journal.pone.0256782.g003

The value of 1 was added at the end of Eq 1 to prevent taking logarithms of zero-counts.

The result of each subsequent pre-processing step can be observed in Fig 4, and the maximum abundance counts in each bracket are shown in Fig 5.

The roles of rarely-occurring species may be influential in a microbiological community, and ultimately decisive with regards to an outcome. In order to preserve the stark contrast in their high abundance compared to the many low-abundance species, an autoscaling transformation *was not applied* to the species counts. If one were applied, then each species would essentially have the same mean and variance in terms of counts, which would render them indistinguishable in subsequent modelling.

Predictive modelling using machine learning

The case studies presented in this paper are categorized, in the ML field, as *classification* problems. A classification model describes a possible mapping between measured samples of raw, input variables ***X***, and corresponding measurements of a target variable ***y*** which are discrete membership (or class) labels. In contrast, a problem dealing with a continuous target ***y*** is known as a *regression* model. The methods presented in this paper will apply exclusively to *classification* and not *regression* problems.

The discussion will start with an introduction of two popular classification models—Random Forests (RFs) and Support Vector Machines (SVMs). However, these relatively simple models will only serve as a baseline of performance (i.e. model accuracy) to compare against. Most of the analysis will be focused on predictive classifiers using Deep Learning (DL) architectures, as well as its optimally-separating variant which we propose as an improved feature extractor.

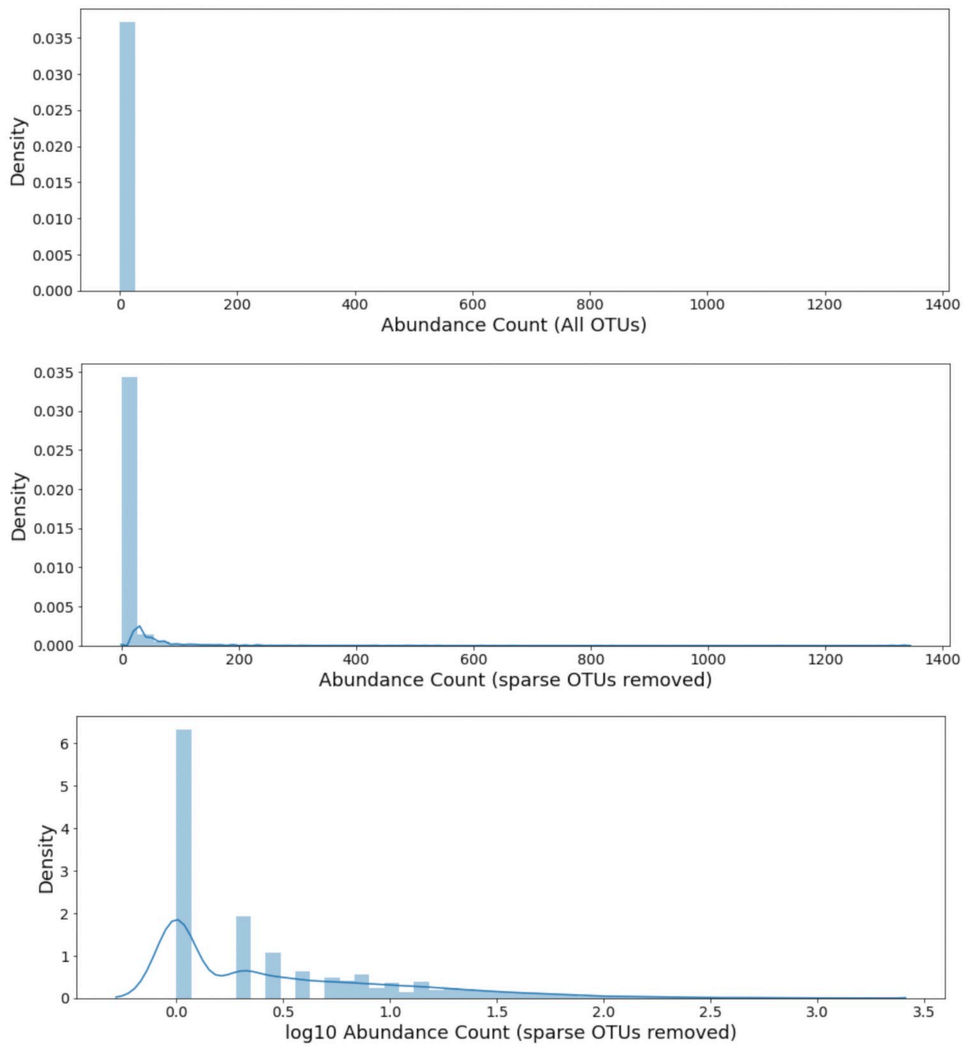


Fig 4. Species abundance counts after each pre-processing transformation. *(Top)* Distribution of counts from all 21721 species; the density is extremely skewed towards the low end. *(Middle)* Distribution of only the top 50 species by sum. *(Bottom)* Distribution of the top 50 species after a log_{10} transformation. Rarer but higher-abundance species are now recognizable.

https://doi.org/10.1371/journal.pone.0256782.g004

Traditional classification models

Two traditional classifiers are used in this work to obtain a baseline model performance, and they are:

1. Random Forests (RFs) [36]
2. Support Vector Machines (SVMs) [37]

RFs are a type of classifier in which the final membership label of each sample is decided by splitting each raw variable conditionally, in a binary manner. For example, one of the first splits in a RF model may be decided using the following set of binary conditions:

- $\boldsymbol{x}_j \geq$ thresh
- $\boldsymbol{x}_j <$ thresh

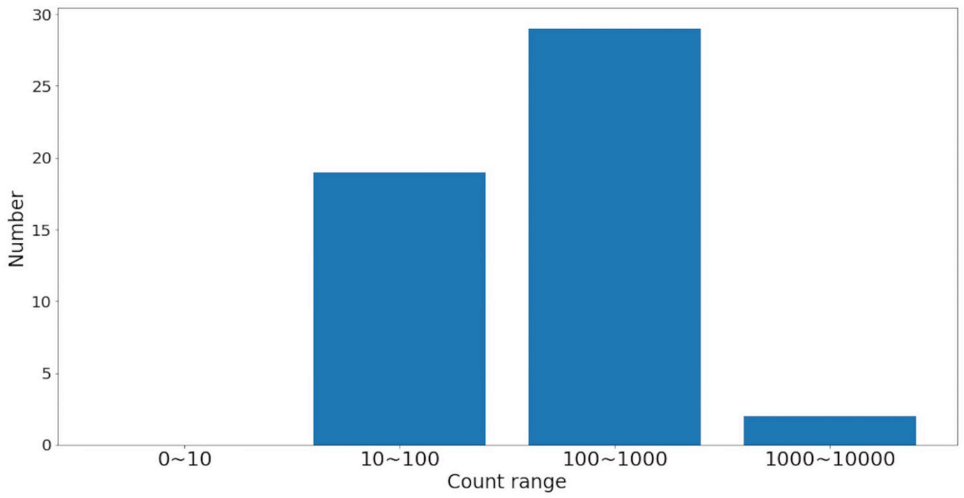


Fig 5. Histogram of maximum abundance counts.

https://doi.org/10.1371/journal.pone.0256782.g005

Here, $\boldsymbol{x}_j$ represents the j^{th} raw variable in the input data, and the threshold is a critical value which further decides how the data samples are classified. Since this process repeats throughout the d total available raw variables, the computational cost of exploring every possible combination becomes too astronomical to be feasible by conventional algorithms. RFs overcome this problem by splitting on a randomly-selected set of variables with each iteration, then *aggregating* across the results obtained from a large number of iterations. This strategy is known as an *ensemble* approach. It is often accompanied by *bootstrapping*, which selects a subsample of all available N data samples with replacement. Therefore, RFs are considered a member of the *bagging* (or *b*ootstrap *agg*regating) family of models.

SVMs are a type of classifier which is popularly used in datasets with a modest number (i.e. tens) of raw variables d [11]. If d is too large, then SVM is susceptible to the *curse of dimensionality* phenomenon like many other ML models. In SVMs, the input data are assumed to take on a variable-space, where the different classes can be distinctly clarified by *separating margins*, which are illustrated in the following Fig 6.

In vanilla SVM, the *separating margin* is a linear hyperplane which is found by maximizing its respective distance to the closest members of each opposing class. Mathematically, this is achieved by minimizing the following Eq 2, known as the *hinge loss*:

$$\underset{w}{min}\ \frac{1}{2}||\boldsymbol{w}||_2 + \sum_{i=1}^{N} max\{0,\ 1 - y^{(i)}\boldsymbol{w}^\top \boldsymbol{x}^{(i)}\} \tag{2}$$

The physical intuitions behind this loss can be interpreted, in words, as:

$$\underset{\text{parameters}}{min} \quad \{\text{inverse of separating margins}\} + \{\text{misclassification rate}\} \tag{3}$$

Note that if the raw data do not conform to a linear structure, such as the example shown in Fig 6, then the vanilla SVM may fail to achieve high training and/or testing accuracy. These scenarios are traditionally tackled by use of the *kernel trick* [38]. This strategy transforms the raw input space into a higher-dimensional space, described by nonlinear basis functions which are determined *a priori*. If the non-linearity structure is well-known, then the kernel trick will efficiently transform the data into a linearly-separable form. Similarly, when the exact non-linear structure is unknown but the degree of non-linearity is low, then an adaptation of the

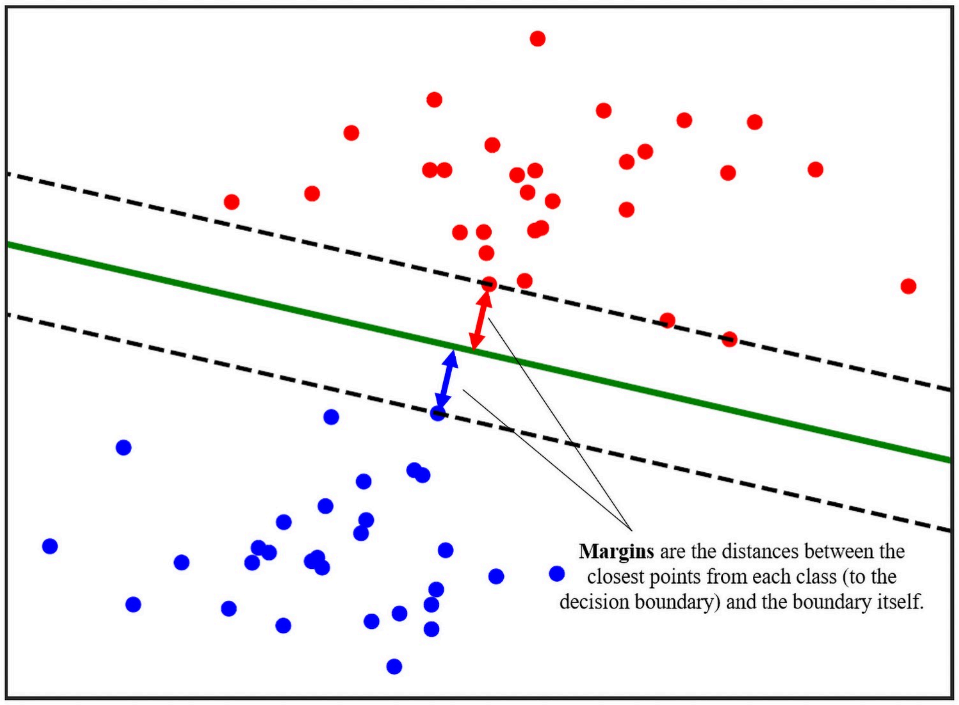


Fig 6. Visualization of a linear separating hyperplane and separating margins in a 2-class SVM model.

https://doi.org/10.1371/journal.pone.0256782.g006

kernel trick known as *Sparse Identification of Nonlinear Dynamics (SinDy)* [39] is equally powerful. However, in the worst-case scenario where both the structure and degree of non-linearity are high and unknown, there is no guarantee that a linearly-separable latent space can be found, even after a lengthy shopping-list of kernels has been exhausted.

This is where our proposed Deep Optimal Feature Learning comes into fruition. The following sections will outline how Deep Learning is generally used as classification models, as well as how we propose to shape the existing Deep Learning objective functions in order to achieve a latent space with improved class separation.

The proposed deep optimal feature extractor

Consider a typical binary classification problem, which has a total of $C = 2$ classes (Fig 7). In many cases, due to non-linear interactions between the raw variables, the classes cannot simply be separated using a linear hyperplane (or separating boundary). Therefore, linear classifiers such as traditional Support Vector Machine (SVM) [37] will often generalize poorly for new samples (i.e. have a low test-set accuracy).

Although this non-linear classification problem can be solved with a sufficient number of iterations using One-Versus-One (OVO) and One-Versus-Rest (OVR) SVM variants [11], finding a latent space which optimally separates the two classes would arguably be a more prudent strategy. Instead of assigning the functional form of the latent space *a priori* by using kernels, we instead propose the use of a Deep Learning model which optimizes the following Eq 4, expressed in words:

$$\begin{aligned} \underset{\text{parameters}}{min} \quad & -\{\text{separating margins}\} + \{\text{misclassification rate}\} \\ \textit{subject to} \quad & \text{separating margins} < \infty \end{aligned} \tag{4}$$

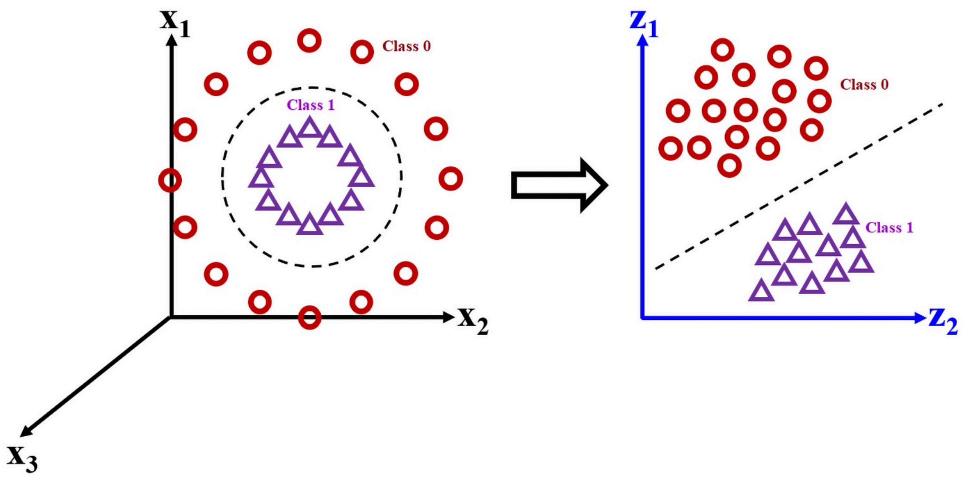


Fig 7. Visualization of a classification problem with a non-linear separation boundary between the two classes. *(Left)* The raw feature-space spanned by raw variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ renders linear separation impossible. *(Right)* A desired latent feature space which optimally separates the two classes. The goal of the DL model is to learn its coordinates, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$.

https://doi.org/10.1371/journal.pone.0256782.g007

Note that this objective function is expressed as a *minimization*, which is standard in Machine Learning literature. Minimizing the negative margins is equivalent to maximizing the margins, under the constraint that the margins remain finite. If this constraint did not exist, then one could arbitrarily choose a latent space where the two classes shown in the right figure of Fig 7 are pushed infinitely far away from each other.

Eq 4 is inspired by the *hinge loss* (Eq 2), the original objective function in the SVM paper by [37]. In Eq 2, the term $\boldsymbol{w}$ refers to the weights or parameters of the SVM model, and the term $\|\boldsymbol{w}\|_2$ is exactly the inverse of the separating margins. Therefore, the inclusion of the term $\|\boldsymbol{w}\|_2$ within the minimization is equivalent to sustaining the finite separating margin constraint outlined in Eq 4. The last summation term represents the total number of misclassified samples, and when visualized on a 2-dimensional plot gives a hinge-like shape (hence its name). The concept of maximizing accuracy and separating margins simultaneously can be extended to DL models. For instance, consider a two-layer neural network as shown in Fig 8.

The square-bracket superscript $^{[L]}$ refers to the L^{th} hidden layer, as conforming to the neural network nomenclature in [25]. The input layer $\boldsymbol{X}$ can be considered the first layer in the network, $\boldsymbol{Z}^{[0]}$. Since the interclass separation occurs between layers $\boldsymbol{Z}^{[1]}$ and $\boldsymbol{y}$, the hinge loss in Eq 2 can be re-written in terms of the neurons and parameters in said layers. Since $\boldsymbol{y} = \boldsymbol{A}^{[2]}(\boldsymbol{w}^{[2]} \boldsymbol{z}^{[1]} + \boldsymbol{b}^{[2]})$ and $\boldsymbol{z}^{[1]} = \boldsymbol{A}^{[1]}(\boldsymbol{w}^{[1]} \boldsymbol{x} + \boldsymbol{b}^{[1]})$, the hinge-loss with respect to this network is:

$$\begin{aligned} \min_{\boldsymbol{w}^{[1]},\, \boldsymbol{w}^{[2]}} \frac{1}{2}||w^{[2]}||_2 \quad &+ \sum_{i=1}^{N} max\Big\{0, 1 - A^{[2]}\big(\boldsymbol{w}^{[2]\top} A^{[1]}(\boldsymbol{w}^{[1]\top}\boldsymbol{x} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}\big) \cdot \\ &\boldsymbol{w}^{[2]\top} \cdot A^{[1]}(\boldsymbol{w}^{[1]\top}\boldsymbol{x} + b^{[1]})\Big\} \end{aligned} \tag{5}$$

Once this neural network has been sufficiently trained (i.e. value of its loss function has decreased past an acceptable threshold), the combination of raw inputs with the largest impact on the outcome can now be identified. Here, we propose a *feature extraction* method which takes direct advantage of the optimal separating latent space discussed previously. The concept behind this method is illustrated in the following Fig 9.

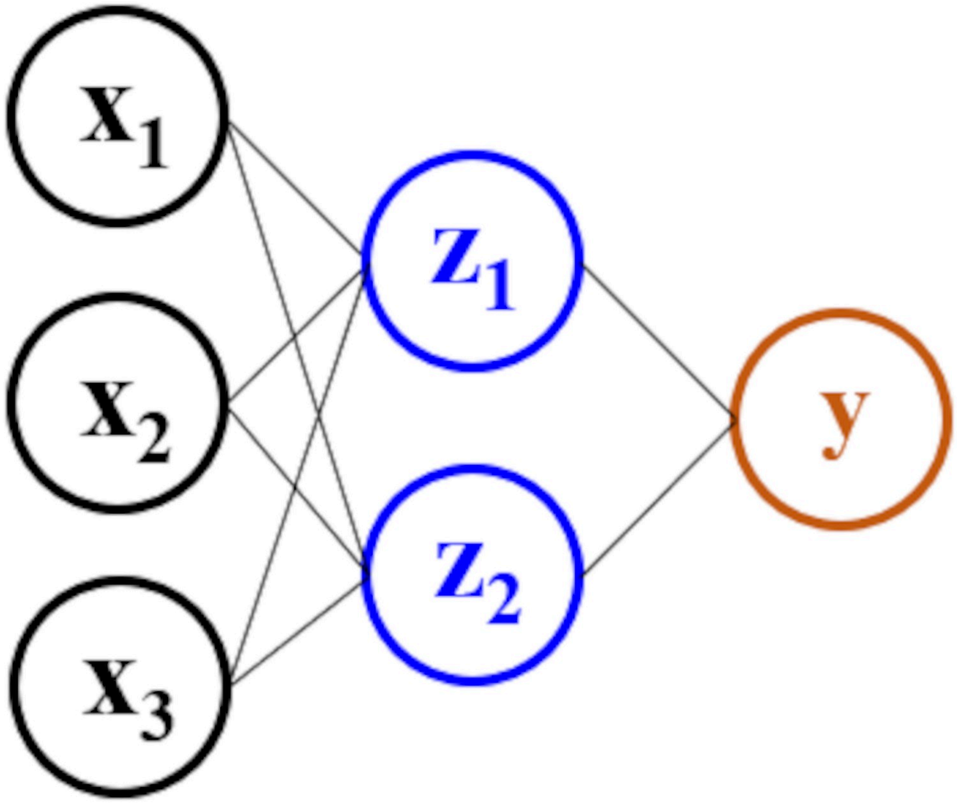


Fig 8. A two-layer neural network with a hinge-loss-like objective function.

https://doi.org/10.1371/journal.pone.0256782.g008

In order to easily visualize this concept, we illustrate it using the special case of a 2-neuron first hidden layer, and point out that the idea applies for any number of neurons $k^{[1]}$. We consider the first hidden layer because it is optimally-separated due to Eq 5. It is also located right after the input layer in the neural network architecture, which means that it is directly affected by the magnitudes of both weights $\boldsymbol{W}^{[1]}$ and input values $\boldsymbol{X}$. In this 2-dimensional hidden layer, $\boldsymbol{z}_1^{[1]}$ and $\boldsymbol{z}_2^{[1]}$ represent the two neurons. The positive and negative classes are separated from one another by a separating margin, which is equal to the Euclidean distance between the two closest points from each opposing class. Moreover, the directional vector of this separating margin can be characterized by $\vec{m}$, which is simply the subtraction of the coordinates between the two closest opposing samples, $\boldsymbol{z}^{(+)}$ (positive class) and $\boldsymbol{z}^{(-)}$ (negative class). Intuitively, $\vec{m}$ represents the direction which best classifies the two opposing classes, as opposed to any other direction found in the latent space spanned by $\boldsymbol{z}_1^{[1]}$ and $\boldsymbol{z}_2^{[1]}$. This property can be realized by projecting all data samples onto the vector $\vec{m}$, observing the class separation on this 1-dimensional line, then repeating the exercise for any other directional vector and comparing the resulting class separations. The elements of $\vec{m}$ can be expressed in the general case (for any value of $k^{[1]}$) as:

$$\vec{m} = [m_1 m_2 \cdots m_{k^{[1]}}] \tag{6}$$

Since the direction of $\vec{m}$ is characterized by its largest elements, we specify a number top_n then extract the same number of top elements (by magnitude). These can top elements can

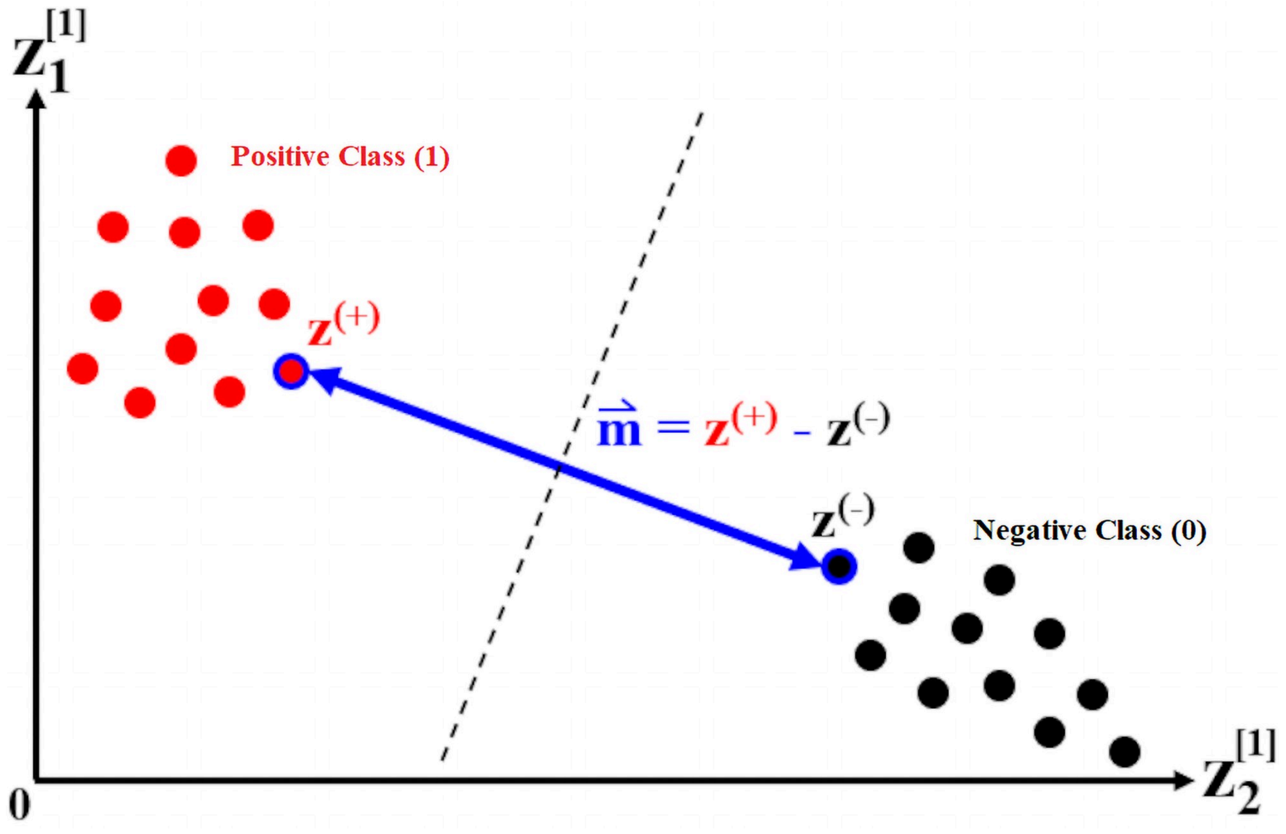


Fig 9. Feature selection based on direction of optimal separation.

https://doi.org/10.1371/journal.pone.0256782.g009

then be collected and expressed as:

$$\underset{top}{\vec{m}} = [m_{top_1} \cdots m_{top_n}] \tag{7}$$

Each of the top elements corresponds to a neuron in the first hidden layer, and hence can be expressed as an activated affine combination of input variables. For instance, if we consider the neuron corresponding to the largest element m_{top_1}, then the following equation holds for any data sample (i):

$$z_{top_1}^{[1](i)} = \boldsymbol{A}^{[1]}(w_{11}^{[1]}x_1^{(i)} + w_{12}^{[1]}x_2^{(i)} + \cdots + w_{1d}^{[1]}x_d^{(i)} + b_{top_1}) \tag{8}$$

From the previous Eq 8, the top_w largest weights (by magnitude) can then be selected from the weight array $[\, w_{11} \quad \cdots \quad w_{1d}\,]$ as:

$$\boldsymbol{w}_{top} = [w_{top_{w1}} \cdots w_{top_w}] \tag{9}$$

The input variables multiplied to these top weights, $[\, \boldsymbol{x}_{top_{w1}} \quad \cdots \quad \boldsymbol{x}_{top_w}\,]$, are therefore the input combinations with most impact on the model output. The rationale behind this weight selection procedure can be best visualized using the basic concepts of matrix multiplication, and is clarified using a simple example shown in Fig 10.

In this example, the first layer values $\boldsymbol{Z}^{[1]}$ are a result of matrix multiplication between the input matrix $\boldsymbol{X}$ and the first hidden layer weight matrix $\boldsymbol{W}^{[1]}$. Suppose the direction of separation $\vec{m}$ has its largest vector element located at position top, and we only consider this top

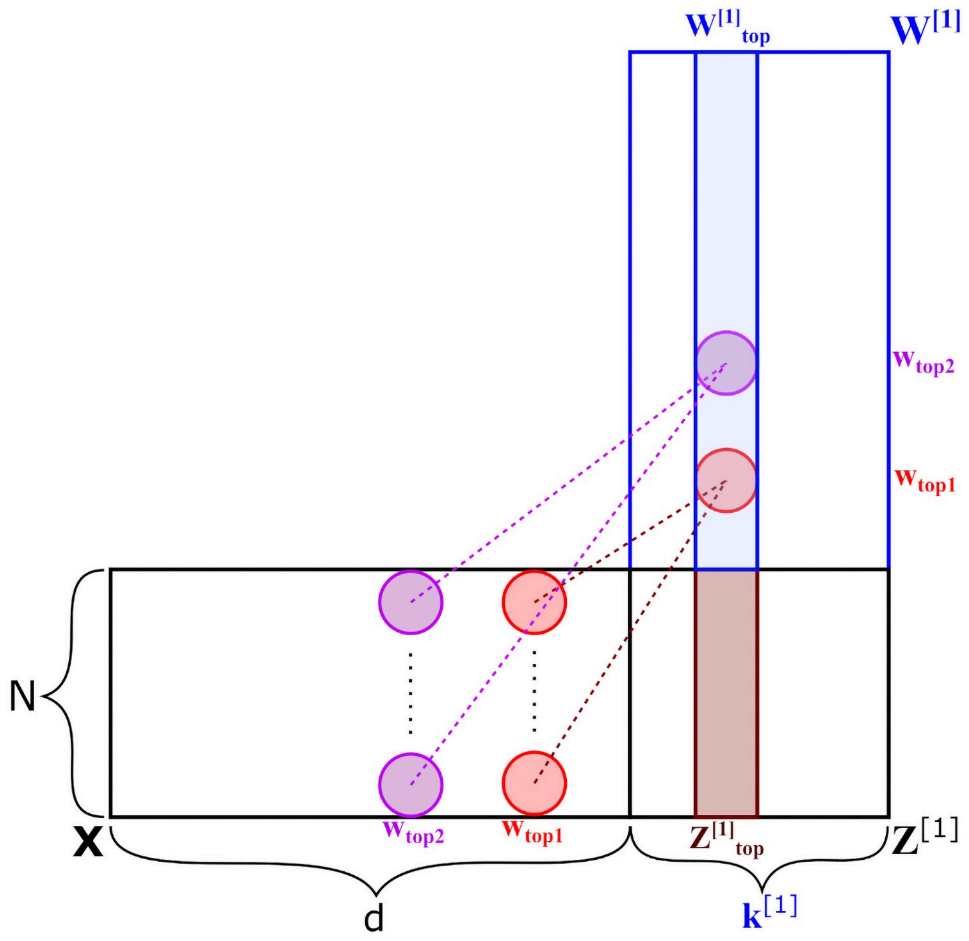


Fig 10. Selection of relevant input variables, by reverse-engineering matrix multiplication.

https://doi.org/10.1371/journal.pone.0256782.g010

element (i.e. $top_n = 1$). The values in column $Z^{[1]}_{top}$ are a direct consequence of matrix multiplication between all d columns in $\boldsymbol{X}$ and all elements in the column $W^{[1]}_{top}$. Therefore, if we now identify the $top_w = 2$ largest elements in this column, their positions $\boldsymbol{w}_{top_1}$ and $\boldsymbol{w}_{top_2}$ correspond to the two most impactful input variables within the set $[\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_d]$.

The entire feature extraction procedure can be summarized as follows. First, train a binary classifier with the loss objective specified in Eq 5, until the loss value falls below an acceptable threshold. Repeat model training over a sufficiently-large number of experiments, such that the results from each random initialization can be later aggregated. Then, perform the following for each model training experiment:

1. Calculate the Euclidean distance matrix between all pairwise samples in the two binary classes. Identify the two opposing-class samples $\boldsymbol{z}^{(+)}$ and $\boldsymbol{z}^{(-)}$ that span the separating margin as the pair with the shortest distance.
2. Calculate $\vec{m}$, the direction of separation, by substracting either the coordinates of $\boldsymbol{z}^{(-)}$ from those of $\boldsymbol{z}^{(+)}$, or vice versa.
3. Calculate the absolute values of all elements in $\vec{m}$, then identify the top_n largest elements based on these absolute values.

4. For each top neuron z_{top_j}, identify the top_w largest weights in the activated affine combination $z_{top_j}^{[1](i)} = \boldsymbol{A}^{[1]}(w_{11}^{[1]}x_1^{(i)} + w_{12}^{[1]}x_2^{(i)} + \cdots + w_{1d}^{[1]}x_d^{(i)} + b_{top_1})$. The combination of input variables $\left[\boldsymbol{x}_{top_{w1}} \quad \cdots \quad \boldsymbol{x}_{top_w} \right]$ multiplied to these top weights are therefore the most impactful features.

In order to reduce variance caused by random initializations of the DL model, this process should be repeated over a large number of experiments. In each experiment, a DL model should be initialized from scratch and trained, and the extracted weights from each experiment should be recorded. Once all experiments are complete, the most impactful weights can then be ranked by majority vote across all experiments. Note that in Step 4 of the previous procedure, the top_w largest weights can be identified based on absolute values, if the goal is to simply extract a set of the most relevant features. However, if a distinction is required between positively and negatively-correlated weights, then the weight ranking should be performed in terms of their unmodified values.

Summary of data analysis steps

The following Fig 11 is a visualization of the data analysis procedure detailed in the previous sections, which transforms raw input data into a set of predictions as well as relevant extracted features.

The pre-processed data is used to train two classification models. The first is a traditional Deep Learning classifier, with the expected *softmax* loss function. The second is the proposed *optimally-separating* Deep Learning classifier, which uses Eq 5 as its loss function in order to improve clarification between the data classes. Two important results from each model—the class (habitat) predictions and extracted features (indicator species)—are compared and contrasted against the results cited in [34].

Deep Learning model implementation

The Deep Learning models were implemented by code written in *iPython 3.7* Jupyter notebooks, using the *PyTorch* package. *PyTorch* was preferred over *Tensorflow* for two reasons. First, the customization of the activation functions and order of computations is less cumbersome in *PyTorch*. Secondly, the proposed optimally-separating cost function in Eq 5 requires an optimization to be performed directly on the neural network parameters. These parameters can be directly obtained in *PyTorch*, by recursively storing the parameters calculated during

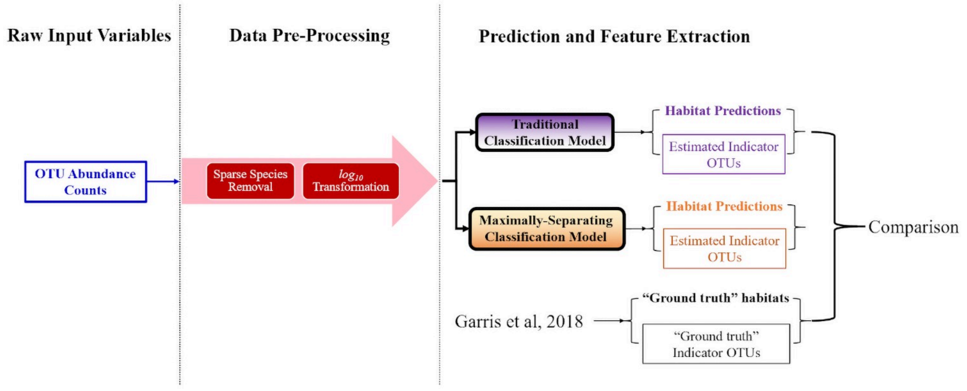


Fig 11. The overall data analysis workflow applied to the Mount Polley case study.

https://doi.org/10.1371/journal.pone.0256782.g011

each epoch then recalling them. In *Tensorflow* and *keras*, these parameters are much more difficult to access and modify.

We will now describe in detail the architecture of each ANN constructed, which can be generalized using a common set of rules. The layers of each ANN can be generalized as follows:

1. **Layer 0**: The input layer, with number of dimensions equal to the number of raw features (in this case, number of species).
2. **Layers 1** $\sim$ $\boldsymbol{(L-1)}$: The hidden layers, with number of neurons in each layer stored in an array.
3. **Layer** $\boldsymbol{L}$: The output layer, with C total neurons and a *softmax* activation for classification.

The neural network therefore has $L+1$ total layers, L layers if the input layer is ignored, and $L-1$ layers if only the hidden layers are counted (i.e. excluding both input and output layers). The sequence of computations in the l^{th} hidden layer, applied to each data sample, can be described as follows:

1. **Affine function**: $\boldsymbol{w}^{[l]\top}\,\boldsymbol{z}^{[l-1]}+\boldsymbol{b}^{[l]}$
2. **Activation function**: $\boldsymbol{z}^{[l]}=\boldsymbol{A}^{[l]}(\boldsymbol{w}^{[l]\top}\,\boldsymbol{z}^{[l-1]}+\boldsymbol{b}^{[l]})$

The affine function is performed on the neuron values obtained from the last layer, $\boldsymbol{z}^{[l-1]}$. The neuron values of the current layer, $\boldsymbol{z}^{[l]}$, is the value of the affine function transformed by the currently desired activation, $\boldsymbol{A}^{[l]}$. This activation is chosen as *ReLU* for all neural networks used in this study. Finally, the output layer (layer L) of each ANN obeys the following sequence of computations:

1. **Affine function**: $\boldsymbol{w}^{[l]\top}\,\boldsymbol{z}^{[l-1]}+\boldsymbol{b}^{[l]}$
2. **Activation function**: $\boldsymbol{A}^{[l]}(\boldsymbol{w}^{[l]\top}\,\boldsymbol{z}^{[l-1]}+\boldsymbol{b}^{[l]})$
3. ***Softmax* transformation**: $\boldsymbol{z}^{[L]}=softmax(\boldsymbol{A}^{[L]}(\boldsymbol{w}^{[L]\top}\,\boldsymbol{z}^{[L-1]}+\boldsymbol{b}^{[L]}))$

At the last (L^{th}) layer, C neurons are present, with each neuron value representing a *logit*. These can be considered unscaled *log*-likelihoods of the data sample belonging to each class. More specifically, *logit* 1 represents the log-likelihood of the sample belonging to Class 1, going all the way up to *logit* C with a similar interpretation. The final *softmax* function transforms all these *logits* into proper probabilities that sum up to 1. Hence, the final array outputted by the *softmax* contains C elements, with each element representing the $0\sim1$ probability that it belongs to class $c\in C$.

In order to achieve a rational compromise between *underfitting* and *overfitting*, each ANN is set to train for a maximum of 1000 epochs, with two overfitting mitigation strategies employed:

1. **Regularization**: An ℓ_2-regularizer is introduced in each layer, which adds the term $\alpha\|\boldsymbol{w}^{[l]}\|_2$ to the cost function (Eq 5) for the weights in all layers.
2. **Early-stopping**: The loss value from Eq 5 is calculated and stored for every epoch. If the loss values have not changed by more than ES_{thresh} over the last n_{ES} epochs, then the training terminates prematurely even if the maximum number of epochs has not been reached.

The regularizer coefficient α is typically set to 0.01. The number of epochs for early-stopping n_{ES} is set to 5, and the early-stopping threshold value ES_{thresh} is set to 10^{-3}. These values may differ slightly for each ANN in order to maintain the underfitting-overfitting balance.

The ANN used for indicator identification in the environmental case study can be visualized in the following Fig 12. The number of raw variables in the input layer is $d=50$. The

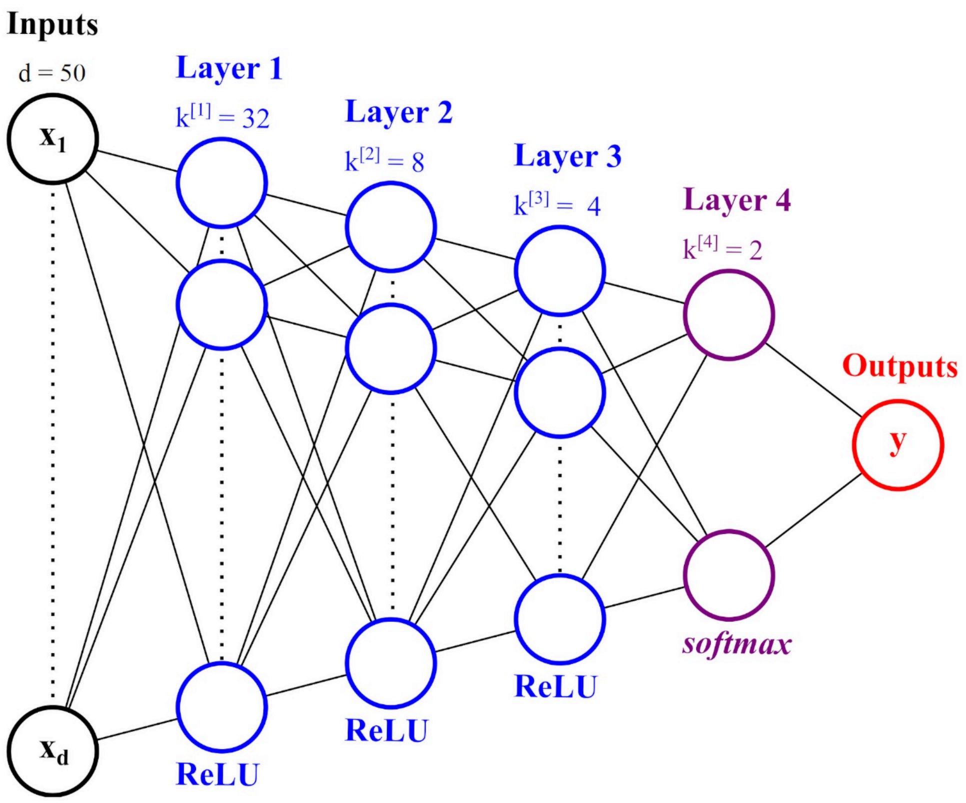


Fig 12. The ANN neuron architecture used in the Mount Polley case study.

https://doi.org/10.1371/journal.pone.0256782.g012

number of neurons in the first hidden layer of the ANN is usually determined by systematic search methods such as [40, 41], or simply specified as a value close to the number of raw variables *d*. Therefore, in this particular case study, we have chosen 32 as the number of neurons in first hidden layer. The second hidden layer uses 8 neurons; this compression serves as a "feature mapping" which extracts only the most relevant latent combinations for classification. Similarly, the third hidden layer uses 4 neurons. Finally, the last hidden layer uses 2 neurons, since this number must be equal to the total number of classes $C = 2$ in order for the *softmax* activation to make sense. Each hidden layer uses a *ReLU* activation, except for the last layer which uses the *softmax* activation required for classification.

Results

The efficacy of the optimally-separating feature extractor is first demonstrated on the well-known benchmark dataset, the Fisher *Iris* [23]. Two Deep Learning models were built and trained with identical architectures. The only difference in these models was the loss function: one used a traditional *softmax* loss whereas the other used the proposed optimally-separating loss. The results show that the optimally-separating model produces much larger separating margins between the binary classes, whilst still achieving perfect test accuracy when compared to the traditional. It is also capable of identifying the same relevant features in the *Iris* problem as those expected from domain knowledge.

The second portion of the results will demonstrate the optimally-separating feature when applied to the environmental monitoring case study. We will show that the keystone microorganisms identified using our proposed method are similar compared to the ones found by [34], in terms of taxonomic composition.

Proof of concept: The Fisher *Iris* dataset

Problem statement. The Fisher *Iris* [23] dataset contains $N = 150$ total samples, with $d = 4$ features:

- $\boldsymbol{x}_1$: Sepal length (cm)
- $\boldsymbol{x}_2$: Sepal width (cm)
- $\boldsymbol{x}_3$: Petal length (cm)
- $\boldsymbol{x}_4$: Petal width (cm)

The three types of *Iris* flowers or outcomes, split evenly with 50 samples each, are:

- y_1: *Iris Setosa*
- y_2: *Iris Versicolor*
- y_3: *Iris Virginica*

Domain knowledge states that the flower types are decided according to the last two features, $\boldsymbol{x}_3$ (Petal length) and $\boldsymbol{x}_4$ (Petal width), whereas the first two features (sepal length and width) have a much less noticeable impact. In order to formulate this problem as a binary classification task, we only consider the first $N = 100$ samples of the dataset, which includes the two outcome classes of *Iris Setosa* (Class 0) and *Iris Versicolor* (Class 1).

For this proof-of-concept, two ANNs are trained—a traditional ANN with the typical *softmax* loss function, and an optimally-separating ANN with the loss function specified in Eq 5. Both ANNs are constructed with the architecture shown in Fig 13, and hyperparameters detailed in Table 1.

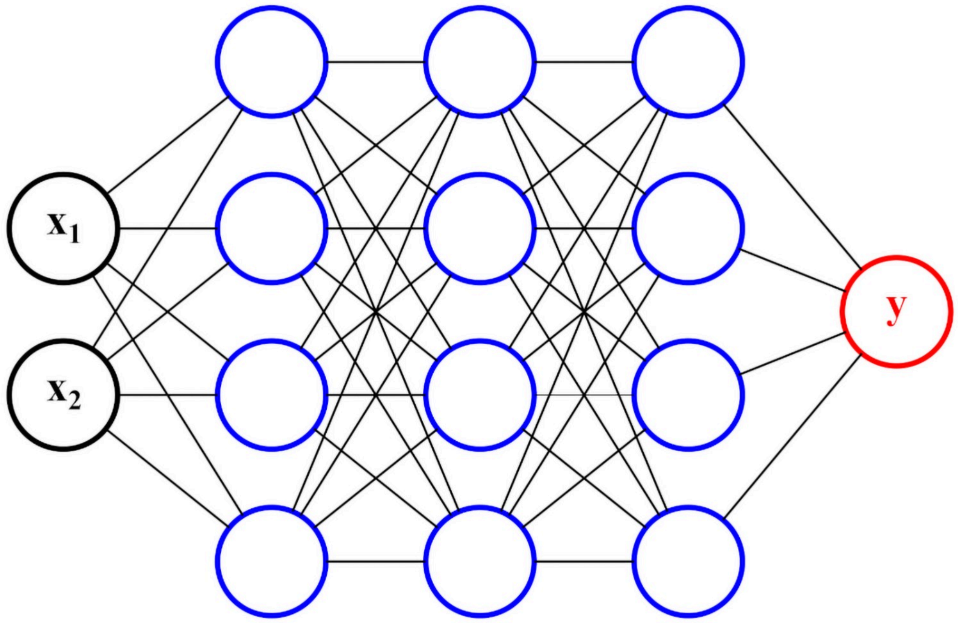


Fig 13. The 4-layer ANN architecture used to classify the Fisher *Iris* dataset.

https://doi.org/10.1371/journal.pone.0256782.g013

Table 1. *Iris* ANN classifier details.

Hyperparameter	Value/Type
Activation in each layer (except last)	*ReLU*
Train:Test ratio	80: 20
Early-Stopping criterion	Average loss over past 5 iterations differs by no more than 1%
Total epochs per experiment	1000
Total experiments	10000

https://doi.org/10.1371/journal.pone.0256782.t001

Note that the training and testing samples, while being randomly selected, is performed such that the ratio of positive to negative class samples is preserved to be the same as in the overall dataset. Moreover, each ANN model is trained over a large number of experiments, since the ANN parameters may initialize in poor locations that converge to high local minima far away from the true global minimum. These result in different top weights, and consequently, different features being extracted. The results are aggregated across all experiments. For each experiment, random training and testing sets are selected with a 80: 20 train:test split ratio.

The results obtained from these two types of loss functions will be compared against one another using the following metrics:

- Model accuracy on the training and testing sets.
- Separating margin between the two classes, i.e. Euclidean (ℓ_2) distance between the two closest opposing samples.

After each of the 10000 ANN models is trained for each case, we first compare the training and testing accuracies of each ANN model, as shown in Table 2.

Since the *Iris* benchmark is a fairly straightforward classification problem, it is no surprise that both ANN models are able to achieve perfect training and testing accuracies, especially when the samples have been reduced to a binary set.

Now that the model generalization accuracy has been ascertained, we compare the interclass separation distances in each hidden layer.

The results in Table 3 show that the positive and negative class clusters in the optimally-separating ANN have been pushed much farther apart, than in the traditional ANN counterpart. This indicates that the hinge-like loss in Eq 5 is working as intended. However, both ANNs

Table 2. ANN training and testing accuracies on the *Iris* dataset.

Model	Average Training Acc. (%)	Average Testing Acc. (%)
Traditional	100	100
Optimally-Separating	100	100

https://doi.org/10.1371/journal.pone.0256782.t002

Table 3. Comparison of separating margins between traditional and optimally-separating ANNs, for the *Iris* classification problem.

Sep. Margin	Traditional			Max. Separating		
Hidden Layer	**1**	**2**	**3**	**1**	**2**	**3**
Avg	1.78	1.87	2.04	6.28	23.27	107.7
Max	2.14	2.37	2.63	13.49	84.24	506.9
Min	1.33	1.36	1.33	1.70	3.45	40.9

https://doi.org/10.1371/journal.pone.0256782.t003

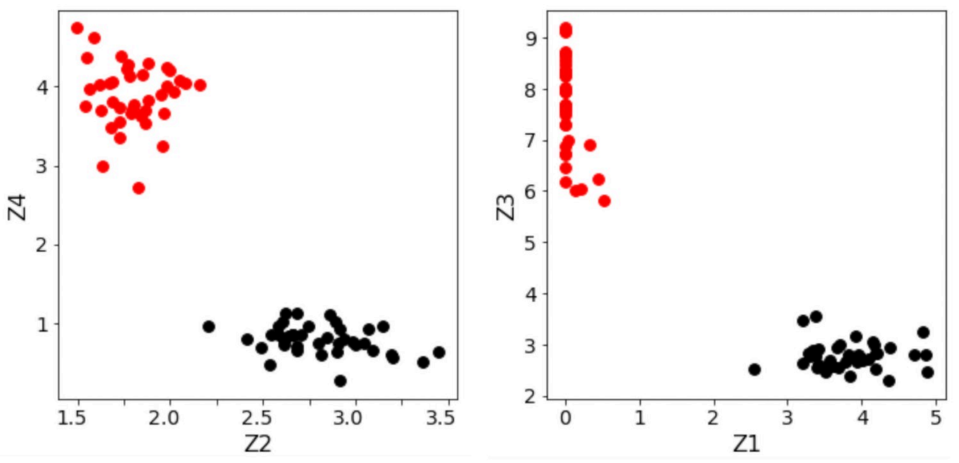


Fig 14. Visualization of data separation within the first ANN hidden layer. *(Left)* Inter-class separation distance in the traditional ANN, between Class 2 (red) samples and Class 1 (black) samples. *(Right)* Inter-class separation distance in the optimally-separating ANN.

https://doi.org/10.1371/journal.pone.0256782.g014

achieve 100% accuracy due to the fact that the two classes are easily separated, as shown in Fig 14.

Finally, we now analyze the feature extraction capabilities of the proposed, optimally-separating ANN. Following the weight-ranking procedure outlined in the previous *Proposed Deep Optimal Feature Extractor* section, we selected $top_n = 2$ largest vector elements, then the $top_w = 2$ largest input weights for each top vector element. The input weights were ranked using their absolute values (rather than their unmodified values), due to no need of differentiation between positively and negatively-correlated weights. The final input variable rankings, i.e. fraction of experiments in which each input variable $[\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \boldsymbol{x}_3 \quad \boldsymbol{x}_4]$ was identified as having the top weights, are reported in Table 4.

The results indicate that the combination of *Petal Length* ($\boldsymbol{x}_3$) and *Petal Width* ($\boldsymbol{x}_4$) input features dominate in terms of deciding the final binary outcome. This agrees with the aforementioned domain knowledge of this well-known dataset. Note that these ranking percentages may change significantly, since we only set a threshold of 10000 experiments due to the limitations of our computing power. However, if a sufficiently large number of experiments could be conducted (ex. 100000 or more), then a more reliable and consistent ranking list could be obtained.

Indicator species case study results

The Fisher *Iris* benchmark results from the previous section indicate that our proposed feature extractor is capable of identifying the correct combination of relevant features impacting the outcome. Now we perform our proposed algorithm on the environmental monitoring dataset,

Table 4. Feature-ranking of the *Iris* dataset, according to our proposed optimally-separating ANN.

Input Variable	Frequency (%)
Sepal Length ($\boldsymbol{x}_1$)	0.0
Sepal Width ($\boldsymbol{x}_2$)	0.0
Petal Length ($\boldsymbol{x}_3$)	54.2
Petal Width ($\boldsymbol{x}_4$)	47.8

https://doi.org/10.1371/journal.pone.0256782.t004

and observe whether the extracted species features are similar to the indicator species identified by the authors of [34].

The first objective of the DL model is to train itself to correctly predict the habitat label of each sample to the best of its ability, given input data in the form of species abundance counts. The second objective of the DL model is to identify *indicator species* with respect to both the undisturbed and disturbed classes. By considering each class as a biocommunity, these keystone species can then be considered main actors which influence the behavior of each biocommunity as a whole. The indicators may subsequently help environmental engineers mitigate the aftermaths of similar spills in the future, by revealing potential ecological shifts in microbial communities.

Finally, after all results are presented, we will compare and contrast the indicator species identified by the Deep Learning feature extractor and the original indicators provided by [34]. The results show that although the two sets of identified indicator species show distinct differences, they also share several similarities. More importantly, the results also indicate that subsequent predictive models can be constructed with higher accuracy when using the new indicator species (from our proposed method) as inputs, as opposed to using the indicators identified by [34].

Predictive modelling results

A total of 21721 species are present in the raw dataset. However, many of these species are sparsely-occurring, meaning their abundance counts are zero across most samples. Considering the extremely small number of samples to start with ($N = 70$), we use a reasonably proportional number of features to prevent *the curse of dimensionality* as much as possible. Therefore, we have chosen to include only the top 50 species, in terms of total abundance counts across all existing samples. Using these abundance counts, we build two ANNs, both with the the architecture specified in Fig 12. One of the ANNs is an optimally-separating one, which uses the loss function prescribed in Eq 5. The other ANN is, of course, a traditional one with a straightforward *softmax* loss. Both traditional and optimally-separating ANNs were trained and tested on the Mount Polley dataset. A comparison of their accuracies (average, maximum, minimum and standard deviations) across 5000 runs (for each case) can be observed in the following Table 5.

Note that the test accuracies are essentially the same across the two types of ANN models. The main reason behind this phenomenon can be observed in the following Figs 15 and 16, which shows two-dimensional plots of the largest separation margins within the layers of each ANN, for the first 4 experiments. Notice that while the separating margins in the traditional ANN (Fig 15) are significantly smaller (in terms of Euclidean distance) compared to those in the optimally-separating ANN (Fig 16), in both cases perfect separation of the binary classes are achieved. Therefore, the increased separating margin does not result in a visible increase in prediction accuracy, as it is expected to do so for cases where the traditional ANN would be unable to cleanly separate the classes.

Table 5. Testing-set accuracy comparison between traditional and optimally-separating ANNs.

Test Acc. (%)	Traditional	Optimally Separating
Avg	98.6	98.7
Max	100.0	100.0
Min	84.6	84.6
Std	2.1	2.1

https://doi.org/10.1371/journal.pone.0256782.t005

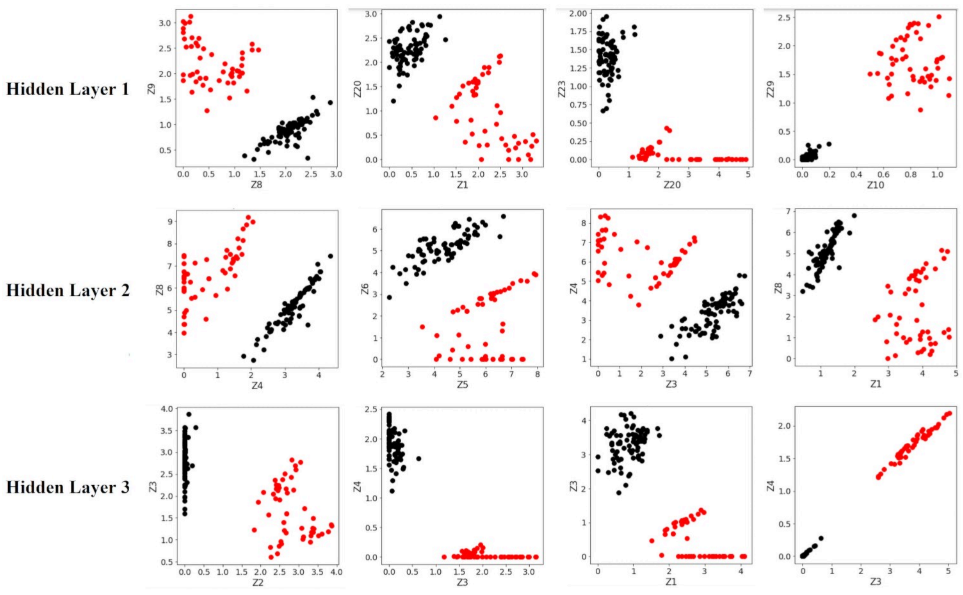


Fig 15. Inter-class separations in the traditional ANN. *Black* samples belong to the undisturbed class. *Red* samples belong to the disturbed class. Only the first 4 out of the total 5000 runs are shown.

https://doi.org/10.1371/journal.pone.0256782.g015

The exact Euclidean distances of the separating margins, averaged across all 5000 runs, are tabulated in the following Table 6. Notice that the increases in margins afforded by the optimally-separating ANN range between 3 to 10 times on average.

After the predictive models—traditional and optimally-separating ANNs—are built and trained adequately, they are then used to extract the indicator species associated with both undisturbed and disturbed habitats. This was performed using the weight-tracing procedure

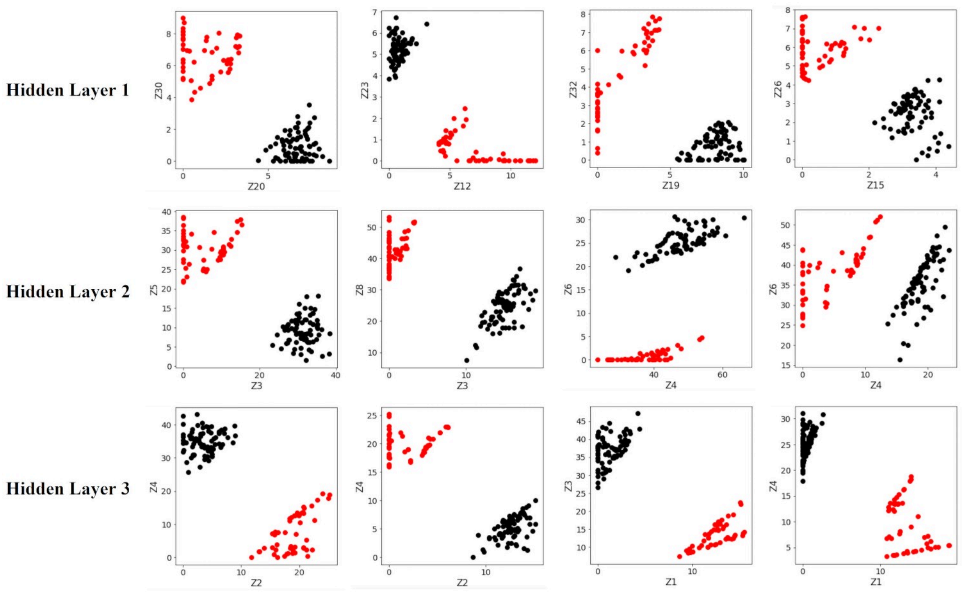


Fig 16. Inter-class separations in the optimally-separating ANN. The separations are noticeably larger than those in Fig 15. *Black* samples belong to the undisturbed class. *Red* samples belong to the disturbed class. Only the first 4 out of the total 5000 runs are shown.

https://doi.org/10.1371/journal.pone.0256782.g016

Table 6. Comparison of separating margins between traditional and optimally-separating ANNs.

Sep. Margin	Traditional			Optimally Separating		
Hidden Layer	**1**	**2**	**3**	**1**	**2**	**3**
Avg	0.88	1.50	1.37	2.77	10.67	15.21
Max	1.93	4.49	4.01	7.99	48.92	73.11
Min	0.36	0.34	0.13	1.52	4.19	6.94

https://doi.org/10.1371/journal.pone.0256782.t006

outlined in the previous *"Proposed Deep Optimal Feature Extractor"* section. The set of indicators identified from the proposed method were used to train 3 types of predictive models—Random Forests, Support Vector Machines, and (traditional) Artificial Neural Networks. This procedure was then repeated using the indicators identified by [34]. The average test-set accuracies across 5000 runs, as well as their standard deviations, are tabulated in the following Table 7.

When the top 50 species (by abundance sum) are used for training, the generalization capability of all three types of predictive models are higher compared to the case where the indicators identified by [34] are used. When the indicators estimated using the optimally-separating feature extractor are used, generalization capability increases slightly for SVM and ANN models, but decreases for the RF model. This is an interesting observation which appears opposite to the expected results between SVMs and RFs. More specifically, the SVM is a linear classifier, whereas the RF is an ensemble method which is able to map out non-linear underlying structure by sheer brute force (i.e. majority votes over a large number of experiments). Since the raw data is highly non-linear in nature, the RF is expected to perform better. Overall, the SVM and ANN test accuracies are the highest when using the estimated indicators, followed by the 50 top species, and worst when using the Garris [34] indicators. These results suggest that in terms of habitat prediction, the set of estimated indicators contain less noisy and/or irrelevant species combinations compared to both Garris [34] and the original dataset.

Indicator species identification

The arguably more important goal of data analytics, in the context of bioinformatics, is to provide insight into the relevant features affecting each specific outcome. In this specific case study concerning the Mount Polley tailings dam breach, we identify indicator species using our proposed strategy, then compare and contrast them to the indicators identified by [34]. The results are shown in the following Table 8:

Two important results in Table 8 are worthy of mention. First, 15 out of the 25 estimated indicators have also been previously been identified by [34]. This implies a healthy extent of overlap between the two vastly different identification methods. Moreover, *Species*1 has been identified as an indicator by both methods. This species is *lithotrophic*, (i.e. it survives by feeding off inorganic material, including solid minerals). It is abundantly present in disturbed habitats that predominantly consist of tailings, which are mineral in origin with little organic

Table 7. Testing-set accuracy comparison between classifiers, using either all species or only indicator species as inputs. Plus-minus value represents one standard deviation.

Test Acc. (%)	50 top species (by sum)	Garris Ind. only	Estimated Ind. only
RF	97.9±1.4	97.36±1.6	96.62±1.7
SVM	98.7±1.5	97.37±1.7	98.72±1.7
ANN	98.7±1.6	97.41±1.8	98.71±1.8

https://doi.org/10.1371/journal.pone.0256782.t007

Table 8. Indicator species identified by our proposed feature extractor, compared to those identified by [34]. The *Frequency* column denotes the number of times each species has been identified as a top weight, divided over all 5000 experiments. The *Garris* column represents whether each species has been identified in the paper [34], and if so, which habitat it belongs to. The maximum and mean abundance counts of each species are also shown, along with the *Class* taxonomy level.

Species ID	Frequency (%)	Garris [34]	Mean(Natural)	Max (Natural)	Mean (Disturbed)	Max (Disturbed)	Taxonomy (Class)
Species39	22.00	N/A	5	28	3	35	Acidobacteria
Species11	10.69	N/A	26	435	12	236	Methanomicrobia
Species1	9.27	Disturbed	3	13	305	1310	Betaproteobacteria
Species17	9.21	Disturbed	2	4	35	218	Flavobacteriia
Species53	7.96	Natural	12	74	2	9	unclassified
Species172	4.62	Natural	8	36	2	7	Methanomicrobia
Species13	3.28	Disturbed	3	13	37	302	Gammaproteobacteria
Species88	3.12	Natural	6	36	2	5	Anaerolineae
Species29	3.08	N/A	6	56	3	9	Cyanobacteria
Species16	3.07	N/A	2	2	24	215	Flavobacteriia
Species42	1.73	Disturbed	5	28	2	3	Methanomicrobia
Species52	1.73	Disturbed	11	64	7	38	Flavobacteriia
Species102	1.70	Disturbed	4	19	28	255	Betaproteobacteria
Species10	1.61	N/A	8	37	6	51	Betaproteobacteria
Species8	1.61	Natural	8	118	9	39	Betaproteobacteria
Species21	1.60	Natural	9	29	4	18	Deltaproteobacteria
Species4	1.58	N/A	2	5	126	488	Bacilli
Species14	1.58	Disturbed	2	9	27	490	Gammaproteobacteria
Species41	1.57	Natural	13	31	3	9	Betaproteobacteria
Species15	1.54	Natural	16	137	3	9	Gammaproteobacteria
Species26	1.51	N/A	4	61	2	13	Betaproteobacteria
Species62	1.51	Natural	9	60	2	12	Deltaproteobacteria
Species9	1.50	N/A	2	4	35	129	Bacilli
Species18	1.49	N/A	10	97	2	5	Cyanobacteria
Species54188	1.49	N/A	8	53	3	12	Chloroplast

https://doi.org/10.1371/journal.pone.0256782.t008

material present. In order to compare these results at a more detailed level, we report the indicator species in terms of the taxonomic levels of *Domain* (Fig 17), *Phylum* (Fig 18), and *Class* (Fig 19). The proportional total number of indicators belonging to each category within each taxonomic level can be visualized in the following Figs 17 through 19. Only the first 3 taxonomic levels were considered, as any level below *class* showed species divisions which were too fine for meaningful comparison. At the *domain* level, our proposed feature extractor identified a similar number of *Bacteria* species, but less *Archaea* species, when compared to the Garris indicators. At the *phylum* level, our method identifies *Proteobacteria* as the predominant type of indicators, which strongly agrees with the Garris results. Both methods also identify *Bacteroidetes* and *Acidobacteria* as indicator types of similar proportionality. On the other hand, our method identifies less *Euryarchaeota* and more *Chloroflexi* indicators. Finally, at the *class* level, our method identifies *Betaproteobacteria* as the predominant type of indicators, which again agrees with the Garris results. However, several indicator species have only been identified by one method alone. For example, Garris et al recognized *Cyanobacteria* and *Chloroplast* species as indicators, while none of them were recognized by our method. In contrast, our method picked up many species such as *Thermoleophilia*, *Methanobacteria*, etc., which were not recognized by Garris et al. Nevertheless, the overall results indicate a satisfactory degree of overlap between our feature extractor and that of [34].

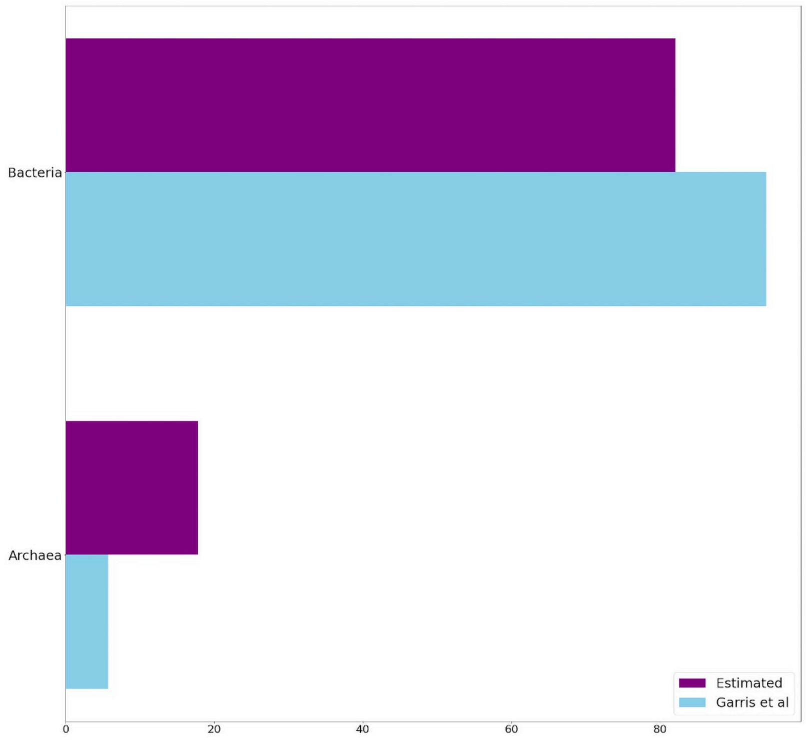


Fig 17. Taxonomic comparison of indicator species at the *domain* level. *Blue* bars represent the indicators identified by Garris et al [34]. *Purple* bars represent the indicators identified by our proposed feature extractor. The horizontal axis represents the percentage of indicators belonging to each species.

https://doi.org/10.1371/journal.pone.0256782.g017

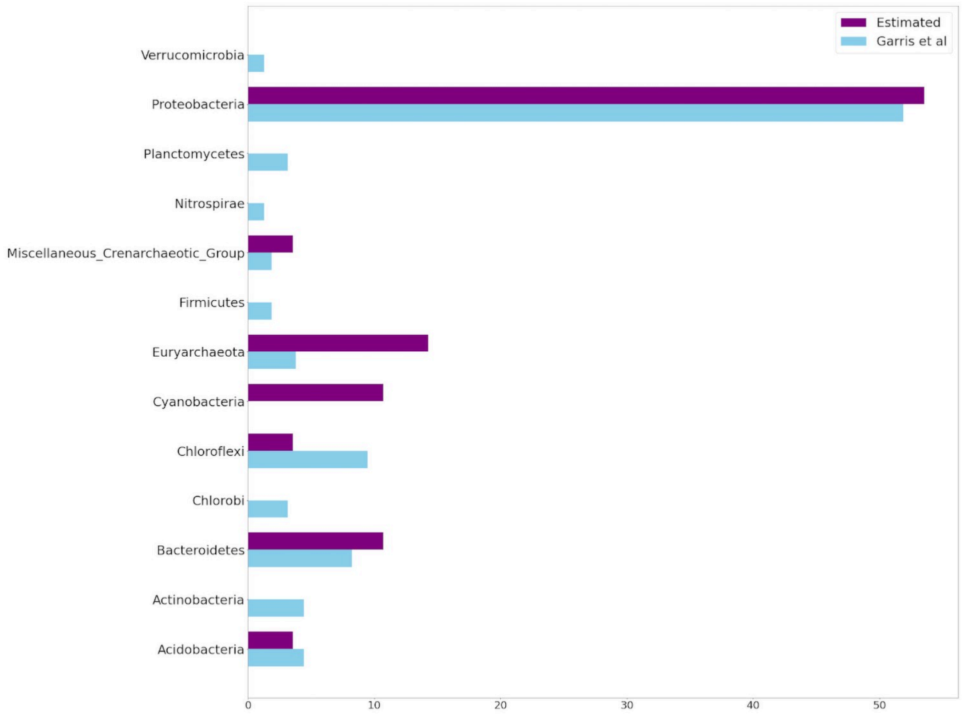


Fig 18. Taxonomic comparison of indicator species at the *phylum* level. *Blue* bars represent the indicators identified by Garris et al [34]. *Purple* bars represent the indicators identified by our proposed feature extractor. The horizontal axis represents the percentage of indicators belonging to each species.

https://doi.org/10.1371/journal.pone.0256782.g018

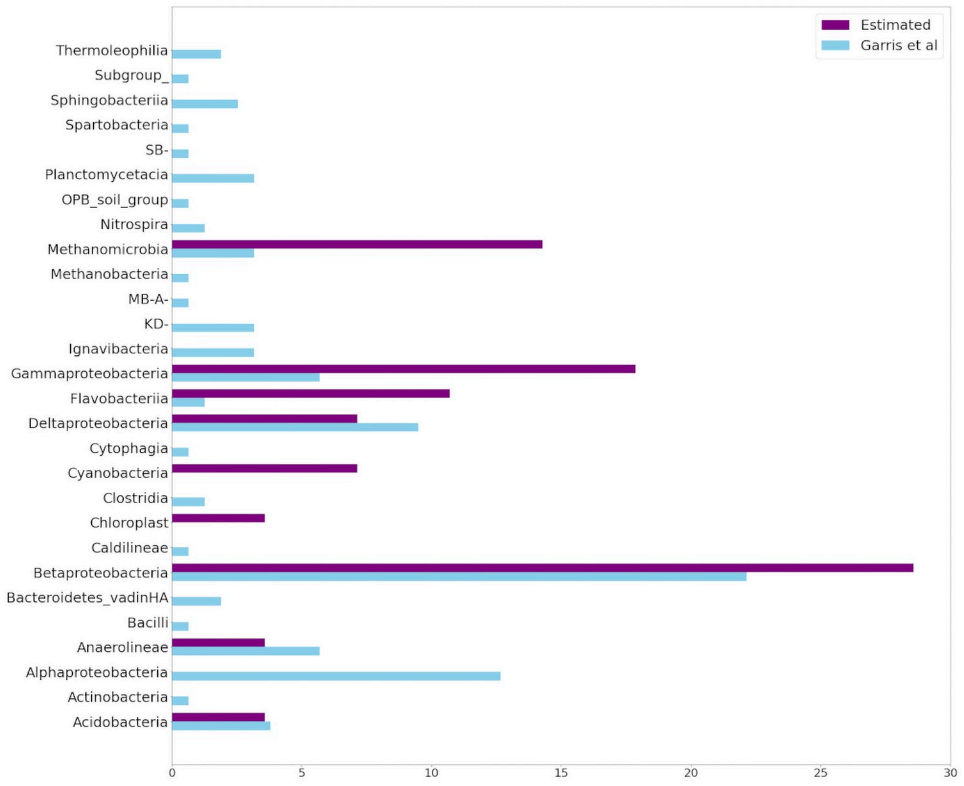


Fig 19. Taxonomic comparison of indicator species at the *class* level. *Blue* bars represent the indicators identified by Garris et al [34]. *Purple* bars represent the indicators identified by our proposed feature extractor. The horizontal axis represents the percentage of indicators belonging to each species.

https://doi.org/10.1371/journal.pone.0256782.g019

Conclusion

In this work, we introduce a novel feature extraction algorithm based on Deep Learning, suited specifically for binary classification problems. The algorithm uses a hinge-like loss function in place of a traditional cross-entropy loss, in order to optimally separate the two distinct classes in the discovered latent spaces. The most relevant input features are extracted by determining the vector which describes the direction of inter-class separation in the latent space, then selecting the largest elements within said vector. We first demonstrate our algorithm on the benchmark *Fisher Iris* classification problem, and show that it is capable at accurate prediction as well as identifying the correct relevant features (i.e. *sepal length* and *sepal width*). We then test our proposed strategy on an environmental monitoring dataset collected from the Mount Polley tailings breach, which includes microorganism species abundance counts from natural and disturbed habitats. The first set of results show that our proposed feature extractor identifies indicator species with similar *domain* and *phylum* taxonomic levels, when compared to those identified in the previous work [34]. The second set of results show that the indicators extracted from our proposed method lead to more accurate habitat predictions, when used in models such as Support Vector Machine or traditional Artificial Neural Network models.

Future plans to improve this work include rigorous sensitivity testing of the proposed feature extractor, as well as expanding its scope and applicability. For instance, the current formulation has only been tested on a relatively small dataset with $N = 70$ samples. It would be interesting to observe whether the same success holds for bigger datasets with thousands of samples, as Deep Learning typically trains better on larger sets [25]. Moreover, a sensitivity study which investigates performance change with respect to hyperparameters top_n and top_w (i.e. the number of top components selected in each feature extraction step) would strengthen the reliability of the algorithm. With respect to the general state of research in the feature extraction field, the main next step is to implement our method on datasets from similar disciplines such as disease (ex. COVID) diagnosis and prevention. These future case studies will produce additional results that compare and contrast the efficacy of our algorithm to well-established such as PCA-Firefly [16], and further clarify whether Deep Learning can indeed provide more intuitive and actionable feature extraction results.

Acknowledgments

The authors would like to thank the authors of [34] for providing relevant data, figures, and other results from their previous study of the Mount Polley incident. Without this information, our work would not have been possible.

Author Contributions

Conceptualization: Yiting Tsai, Susan A. Baldwin.

Data curation: Yiting Tsai, Susan A. Baldwin.

Formal analysis: Yiting Tsai, Susan A. Baldwin.

Investigation: Yiting Tsai.

Methodology: Yiting Tsai, Susan A. Baldwin.

Resources: Yiting Tsai, Susan A. Baldwin.

Software: Yiting Tsai.

Supervision: Susan A. Baldwin, Bhushan Gopaluni.

Validation: Yiting Tsai.

Visualization: Yiting Tsai.

Writing – original draft: Yiting Tsai.

Writing – review & editing: Yiting Tsai, Susan A. Baldwin, Bhushan Gopaluni.

References

1. Rajput DS, Basha SM, Xin Q, Gadekallu TR, Kaluri R, Lakshmanna K, et al. Providing diagnosis on diabetes using cloud computing environment to the people living in rural areas of India. Journal of Ambient Intelligence and Humanized Computing. 2021; p. 1–12.
2. Dufrêne M, Legendre P. Species assemblages and indicator species: the need for a flexible asymmetrical approach. Ecological monographs. 1997; 67(3):345–366. https://doi.org/10.1890/0012-9615(1997)067%5B0345:SAAIST%5D2.0.CO;2
3. Podani J, Csányi B. Detecting indicator species: Some extensions of the IndVal measure. Ecological Indicators. 2010; 10(6):1119–1124. https://doi.org/10.1016/j.ecolind.2010.03.010
4. Penczak T. Fish assemblage compositions after implementation of the IndVal method on the Narew River system. Ecological modelling. 2009; 220(3):419–423. https://doi.org/10.1016/j.ecolmodel.2008.11.005
5. Antonelli L, Foata J, Quilichini Y, Marchand B. Influence of season and site location on European cultured sea bass parasites in Corsican fish farms using indicator species analysis (IndVal). Parasitology research. 2016; 115(2):561–568. https://doi.org/10.1007/s00436-015-4772-9 PMID: 26446088
6. Cox MA, Cox TF. Multidimensional scaling. In: Handbook of data visualization. Springer; 2008. p. 315–347.

7. Shaw PJ. Multivariate statistics for the environmental sciences. Wiley; 2009.
8. Legendre P, Legendre L. Numerical Ecology, Volume 24, (Developments in Environmental Modelling). Elsevier Science Amsterdam, The Netherlands; 1998.
9. Likas A, Vlassis N, Verbeek JJ. The global k-means clustering algorithm. Pattern recognition. 2003; 36 (2):451–461. https://doi.org/10.1016/S0031-3203(02)00060-2
10. Abdi H, Williams LJ. Principal component analysis. Wiley interdisciplinary reviews: computational statistics. 2010; 2(4):433–459. https://doi.org/10.1002/wics.101
11. Bishop CM. Pattern Recognition and Machine Learning. vol. 1. Springer; 2006.
12. Ng AY. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In: Proceedings of the twenty-first international conference on Machine learning. ACM; 2004. p. 78.
13. Han H, Guo X, Yu H. Variable selection using mean decrease accuracy and mean decrease gini based on random forest. In: Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on. IEEE; 2016. p. 219–224.
14. Reddy GT, Reddy MPK, Lakshmanna K, Kaluri R, Rajput DS, Srivastava G, et al. Analysis of dimensionality reduction techniques on big data. IEEE Access. 2020; 8:54776–54788. https://doi.org/10.1109/ACCESS.2020.2980942
15. Hilario M, Kalousis A. Approaches to dimensionality reduction in proteomic biomarker studies. Briefings in bioinformatics. 2008; 9(2):102–118. https://doi.org/10.1093/bib/bbn005 PMID: 18310106
16. Gadekallu TR, Khare N, Bhattacharya S, Singh S, Reddy Maddikunta PK, Ra IH, et al. Early detection of diabetic retinopathy using PCA-firefly based deep learning model. Electronics. 2020; 9(2):274. https://doi.org/10.3390/electronics9020274
17. Tenenbaum JB, De Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. science. 2000; 290(5500):2319–2323. https://doi.org/10.1126/science.290.5500.2319 PMID: 11125149
18. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. science. 2000; 290(5500):2323–2326. https://doi.org/10.1126/science.290.5500.2323 PMID: 11125150
19. Maaten Lvd, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008; 9 (Nov):2579–2605.
20. McInnes L, Healy J, Melville J. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:180203426. 2018;.
21. Sonnewald M, Dutkiewicz S, Hill C, Forget G. Elucidating ecological complexity: Unsupervised learning determines global marine eco-provinces. Science Advances. 2020; 6(22):eaay4740. https://doi.org/10.1126/sciadv.aay4740 PMID: 32523981
22. De Ridder D, Kouropteva O, Okun O, Pietikäinen M, Duin RP. Supervised locally linear embedding. In: Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003. Springer; 2003. p. 333–341.
23. Fisher RA. The use of multiple measurements in taxonomic problems. Annals of eugenics. 1936; 7 (2):179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
24. Hofmann T, Schölkopf B, Smola AJ. Kernel methods in machine learning. The annals of statistics. 2008; p. 1171–1220.
25. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016.
26. Kratsios A. The universal approximation property: Characterizations, existence, and a canonical topology for deep-learning. arXiv preprint arXiv:191003344. 2019;.
27. Winkler DA, Le TC. Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR. Molecular informatics. 2017; 36(1-2):1600118. https://doi.org/10.1002/minf.201600118
28. Gadekallu TR, Khare N, Bhattacharya S, Singh S, Maddikunta PKR, Srivastava G. Deep neural networks to predict diabetic retinopathy. Journal Of Ambient Intelligence and Humanized Computing. 2020; p. 1–14.
29. Segovia F, Górriz J, Ramírez J, Martínez-Murcia FJ, García-Pérez M. Using deep neural networks along with dimensionality reduction techniques to assist the diagnosis of neurodegenerative disorders. Logic Journal of the IGPL. 2018; 26(6):618–628. https://doi.org/10.1093/jigpal/jzy026 PMID: 30532642
30. Balamurugan M, Nancy A, Vijaykumar S. Alzheimer's disease diagnosis by using dimensionality reduction based on knn classifier. Biomedical and Pharmacology Journal. 2017; 10(4):1823–1830. https://doi.org/10.13005/bpj/1299
31. Gang P, Zhen W, Zeng W, Gordienko Y, Kochura Y, Alienin O, et al. Dimensionality reduction in deep learning for chest X-ray analysis of lung cancer. In: 2018 tenth international conference on advanced computational intelligence (ICACI). IEEE; 2018. p. 878–883.

32. Nguyen NP, Warnow T, Pop M, White B. A perspective on 16S rRNA operational taxonomic unit clustering using sequence similarity. NPJ biofilms and microbiomes. 2016; 2(1):1–8. https://doi.org/10.1038/npjbiofilms.2016.4 PMID: 28721243

33. Callahan BJ, McMurdie PJ, Holmes SP. Exact sequence variants should replace operational taxonomic units in marker-gene data analysis. The ISME journal. 2017; 11(12):2639–2643. https://doi.org/10.1038/ismej.2017.119 PMID: 28731476

34. Garris HW, Baldwin SA, Taylor J, Gurr DB, Denesiuk DR, Van Hamme JD, et al. Short-term microbial effects of a large-scale mine-tailing storage facility collapse on the local natural environment. PloS one. 2018; 13(4). https://doi.org/10.1371/journal.pone.0196032 PMID: 29694379

35. Petticrew EL, Albers SJ, Baldwin SA, Carmack EC, Déry SJ, Gantner N, et al. The impact of a catastrophic mine tailings impoundment spill into one of North America's largest fjord lakes: Quesnel Lake, British Columbia, Canada. Geophysical Research Letters. 2015; 42(9):3347–3355. https://doi.org/10.1002/2015GL063345

36. Breiman L. Bagging predictors. Machine learning. 1996; 24(2):123–140. https://doi.org/10.1007/BF00058655

37. Cortes C, Vapnik V. Support-vector networks. Machine learning. 1995; 20(3):273–297. https://doi.org/10.1007/BF00994018

38. Shawe-Taylor J, Cristianini N, et al. Kernel methods for pattern analysis. Cambridge university press; 2004.

39. Kaiser E, Kutz JN, Brunton SL. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. Proceedings of the Royal Society A. 2018; 474(2219):20180335. https://doi.org/10.1098/rspa.2018.0335 PMID: 30839858

40. Alvarez JM, Salzmann M. Learning the number of neurons in deep networks. Advances in Neural Information Processing Systems. 2016; 29:2270–2278.

41. Doukim CA, Dargham JA, Chekima A. Finding the number of hidden neurons for an MLP neural network using coarse to fine search technique. In: 10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010). IEEE; 2010. p. 606–609.