UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

DIVERSITY METRICS IN DEEP LEARNING EMBEDDINGS

Autor: Pablo Prol Prieto

Directors: Nahuel Statuto, Santiago Seguí
Realitzat a: Facultat de Matemàtiques i Informàtica

Barcelona, June 9, 2024

Contents

1 Introduction **1**

2 Fundamentals of deep learning **5**
2.1 Structure of Feedforward Neural Networks 5
2.2 Training Feedforward Neural Networks 8

3 Diversity in Deep Learning **17**
3.1 Ecological Diversity and Hill Numbers 19
3.2 Vendi Score 23
3.2.1 Cousins of the Vendi Score 26

4 Empirical Study **31**
4.1 Hypothesis and discussion of the results 33

5 Conclusions **39**

Bibliography **41**

Appendix **45**
Description of the Used Datasets 45
Additional Plots 46

Abstract

This project embarks on an in-depth exploration of diversity metrics within the realm of deep learning embeddings, under the mentorship of Nahuel Statuto and Santiago Seguí. Our work begins with an introductory overview of our objectives, which were to study the possibility of using a neural network which has been trained for a task to perform well enough in a different task for which it has not been trained, and to approach this study from the perspective of diversity in neural network embeddings. Finally delving into machine learning and the role of diversity in this field. We also lay the foundational knowledge of deep learning, covering the architecture and training of feedforward neural networks. This includes a detailed examination of artificial neurons, their organization into layers, the incorporation of activation functions to model nonlinear relationships, and a deep explanation of the training process, including loss functions, the feedforward process, gradient descent, and backpropagation, explaining how these elements minimize the loss and improve the model's performance.

We then shift the focus to the concept of diversity, first in machine learning, highlighting the limitations of existing metrics and introducing the Vendi Score as a promising alternative. Then, delving into the domain of ecology, we discuss the treatment of diversity in this field, presenting key metrics and a set of properties aligned with the intuition for diversity and considered fundamental by ecologists for diversity metrics that account for species similarity.

Following this, we define the Hill numbers, an ecological diversity metric that serves as the foundation for the Vendi Score and despite being a powerful metric, its main drawback is that it does not take into account similarity between species. After addressing the Hill numbers' limitations, we present the Vendi Score as the extension of the Hill number of order 1, alongside necessary lemmas to demonstrate its adherence to desired properties. Recognizing the Vendi Score's susceptibility to imbalanced data classes, we introduce its extensions, the Cousins of the Vendi Score, which mitigate this issue by extending the Vendi Score to the rest of the Hill numbers, while acknowledging remaining challenges.

The concluding chapter presents empirical studies conducted to study our objectives in a practical setting. Our findings reveal no explicit correlation, other than the diversity of embeddings being directly related to a classifier's number of classes. We conclude that the potential for repurposing a neural network to excel in an untrained task is minimal, highlighting how the relationship between diversity metrics and the effectiveness of deep learning models is more complex than it seems.

2020 Mathematics Subject Classification. 68T07, 68R12.

Chapter 1

Introduction

Artificial intelligence is the field of computer science that studies the development of algorithms and systems which can perform tasks that are usually carried out by humans. Machine learning (ML) is a branch of artificial intelligence and the science of credit assignment: finding patterns in observations that predict the consequences of actions and help to improve future performance [7]. Originally, it was invented to solve tasks that are hard or unfeasible for traditional algorithms (tasks which can not be carried out just by following a set of programmed instructions). For instance, image and speech recognition, natural language processing, and prediction tasks in finance, healthcare or any other field where large amounts of data are available.

Machine learning models learn from data. In supervised learning, the learning process involves training a model on a dataset, which is a collection of examples, each containing a set of features and a target label. The model's goal is to learn a mapping from features to targets that generalizes well to new examples. This goal is established by a loss function, chosen according to the task. To ensure the generalization capability of a model which is to be trained, it is common to split the dataset into two parts: a training set and a test or validation set. The training set is used to train the model, while the test set is used to evaluate the model's performance, without letting the model learn from it. The learning process is guided by the loss function, which measures the discrepancy between the model's predictions for the targets and the actual targets in the training data. The model's parameters are adjusted to minimize this loss function, typically using iterative optimization algorithms.

However, supervised learning is just one method of machine learning. There are other methods that do not involve a labeled dataset. For instance, in unsupervised learning, models are trained on data without explicit labels, aiming to find patterns or structures within the data itself. Common tasks in unsupervised learning

include clustering, where the goal is to group similar examples together, and dimensionality reduction, which seeks to simplify the data without losing important information. Another method is reinforcement learning, where a learner learns to make decisions by performing actions in an environment to achieve some goals. The agent receives feedback in the form of rewards or penalties and learns to maximize cumulative rewards over time. This method is particularly useful for sequential decision-making problems and has been successfully applied in areas such as game playing and robotics.
Among the various machine learning models, neural networks have gained significant popularity over the last decades, due to their ability to learn complex patterns in data. A neural network is a model inspired by the human brain, consisting of interconnected nodes or "neurons". Neural networks have led to the emergence of deep learning, a subfield of machine learning that focuses on training these types of models. A neural network consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment, other neurons get activated through weighted connections from previously active neurons [8]. These models have achieved state-of-the-art performance on a wide range of tasks, outperforming traditional machine learning models in many cases.

Diversity is a concept which has been studied for many years in various fields such as ecology, sociology, and economics, as it is an intrinsic property of many complex systems. Usually, diversity refers to the grade of presence of a multitude of distinct elements within a system. This principle is a fundamental characteristic of both social constructs and natural ecosystems.
In ecology, diversity is a measure of the health of biological systems, as a diverse ecosystem is more resilient to changes and disturbances as well as more capable of sustaining a wide range of life forms. Diversity in ecology is often measured by the variety of species, genetic variability and heterogeneity.
Translating this concept into the realm of technology, particularly to the field of machine learning, diversity is a relevant criterion in many areas, such as dataset curation, generative modeling, reinforcement learning, active learning and decoding algorithms.
A lack of diversity in datasets and models can limit the usefulness of machine learning in many critical applications, for instance scientific discovery [20]. Diversity is also used in recommender systems [11] to ensure that users are exposed to a wide range of content, rather than being stuck in a filter bubble. It is therefore important to be able to measure diversity in machine learning.
Narrowing down to the field of deep learning embeddings, diversity is a crucial aspect in the evaluation of the quality of embeddings, as it is essential that the

embeddings capture a wide range of patterns in the data, or in other words, that they are sufficiently diverse.
In many cases, when using machine learning models for a specific task , we come across the need of using the same models for different tasks for which they have not been trained specifically. In this context, we are interested in the ability of the models for retaining the diversity of the original training examples. In this project we will study ways to approach this issue, focusing in neural networks and the Vendi Score, a diversity metric in machine learning introduced in [20] which presents several advantages over other existing diversity metrics in machine learning.

Machine learning has fascinated me ever since I discovered its capabilities and the wide range of applications it has. Deep learning scales up this capabilities even further, especially the most recent generative models, which keep improving day by day. Deep learning interested me even more when I understood how it works, the mathematics behind it, how calculus and linear algebra are fused together to allow models to learn from huge amounts of data in an efficient manner. When my advisors proposed me to work on diversity in deep learning embeddings and after I started to do some research, I quickly got interested in the topic, as it has not yet been studied in depth and it is still being developed (as a matter of fact, the Vendi Score was introduced in 2022 [20] and its extension, the Cousins of the Vendi Score, was introduced during the development of this project [21]).
I am very grateful to my advisors for proposing me this project, as it has allowed me to learn about diversity in machine learning in general, and to deepen my knowledge in deep learning as a whole, which is a field I am very passionate about. I am also grateful to them for the guidance they have provided me throughout the development of this project.

We start by introducing the fundamentals of deep learning in Chapter 2, including the structure of neural networks and the training process for these type of models. Most of the concepts in Chapter 2 are required in order to understand the following concepts discussed in the project. In Chapter 3, we explore the concept of diversity in deep learning, its importance, the existing metrics and their limitations, focusing on the Vendi Score, which we present along with its extension to the family of Vendi Scores, after a brief review of ecological diversity and the metrics used in this field for measuring diversity, with a focus on the Hill numbers, which are the foundation for the Vendi Score.
In Chapter 4, we present the experiments carried out by ourselves and their respective results, along with a discussion of the results. Finally, in Chapter 5, we

present the conclusions drawn from the results and the conclusions of the project as a whole. Moreover, we discuss possible future work that can be done in this field.

Chapter 2

Fundamentals of deep learning

We already introduced machine learning and mentioned neural networks as the model that gives arise to deep learning. From now on, this project will focus on deep learning and therefore on neural networks.
In modern deep learning, there are many types of neural networks and new types are being developed constantly. However, to limit the scope of this project, we will focus on feedforward neural networks, which are the basic type of neural networks and the baseline for more complex models. In this chapter we explain the structure of feedforward neural networks (which is a specific type of neural networks), from the basic building block, the artificial neuron. We explain how these neurons are grouped into layers, forming the multilayer perceptron and concluding with the concept of activation functions, in which lies the non-linearity of the neural network. We also explain in detail the training process of neural networks, starting by explaining the loss or objective functions, which determine what the neural network will minimize throughout its learning process, the feed forward process, which is the computation of the output of the neural network given an input vector, and concluding with an explanation of the gradient descent algorithm and how it is used to minimize the loss function in the backpropagation algorithm, in which the parameters of the neural network are updated.

2.1 Structure of Feedforward Neural Networks

The purpose of this section is to explain how are feedforward neural networks structured from the artificial neurons to the whole network, going through the concept of activation functions and the layers of the network.
Let us start by defining the artificial neuron.

Artificial Neuron

We can think of artificial neurons as simple functions, which take an input vector and produce an output through a weighted sum of the input vector and a bias term.

Definition 2.1. *Given an input vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, an **artificial neuron** is a function $y : \mathbb{R}^n \to \mathbb{R}$ such that:*

$$y(x) = \sum_{i=1}^{n} w_i x_i + b,$$

where $w = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$ is the weight vector of the neuron and $b \in \mathbb{R}$ is its bias term.

Multilayer Perceptron

Now that we know what is the basic unit of a neural network, we can combine multiple artificial neurons to form an **artificial neural network** or **multilayer perceptron** (MLP). However, as we have defined it, a single neuron can only model linear relationships between the input and the output. Therefore, if we were to stack multiple neurons in a single layer and multiple layers in a network, the network would essentially be a really complex linear model. For that reason, we now introduce the concept of **activation functions**, which allow the network to model non-linear relationships between the input and the target. An **activation function** ϕ is a non-linear function $\phi : \mathbb{R} \to \mathbb{R}$, which takes a single input z, the weighted sum of the inputs to the neuron, and produces an output $a = \phi(z) = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right)$, which we call the **activation** of the neuron.
Here are some of the most common examples of activation functions.

Example 2.2. Rectified Linear Unit (ReLU):

$$\phi(z) = \max(0, z).$$

It has become the most popular activation function in recent years due to its simplicity and computational efficiency, since it simplifies the computation of the gradient during the training process which we will explain in the next section.

Example 2.3. Softmax Function:

$$\phi(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

for a vector $z = (z_1, z_2, \ldots, z_K)$ where each z_i is the output of a neuron. The vector $\phi(z) = (\phi(z_1), \phi(z_2), \ldots, \phi(z_K))$ can be interpreted as a probability distribution

over K different possible outcomes. Hence, the softmax function is often used in the output layer of a neural network trained for K-class classification tasks. Namely, the output of such a network is interpreted as a vector of probabilities for each class being the correct one.

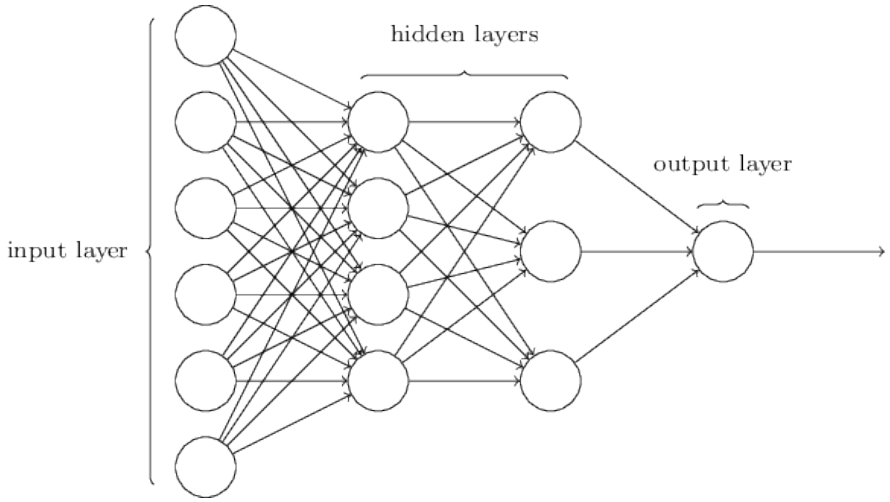

Figure 1: **Structure of a feedforward neural network.** [1] This type of neural network is what we call fully connected, which means that each neuron in a layer is connected to each neuron in the next layer, and thus each neuron in a layer receives input from each neuron of the previous layer and has a weight parameter for the connection with each of the neurons of the previous layer.

Let us use the feedforward fully connected neural network from Figure 1 to exemplify the structure of this type of neural network. The first column of neurons (what we call input layer) is essentially weighing the input evidence. With respect to the neurons in the second and third layers (which we call hidden layers), each of them is creating an activation by weighing up the results from the previous layer of outputs. In this way, a neuron in the second layer, for instance, can create an activation at a more complex and more abstract level than neurons in the first layer. And even more complex activations can be made by the neuron in the fourth layer (which we call output layer). In this way, a many-layer network of perceptrons can engage in sophisticated activation making. The dimension (e.g. number of neurons) of the input layer always corresponds to the dimension of the input vector. Every layer which is neither the input layer nor the output layer is called a hidden layer, and the dimension of hidden layers can be tweaked to find the best model for the task at hand. This task also determines the dimension of the output layer, which is usually the number of classes in a classification task or the number of dimensions of the target in a regression task.

2.2 Training Feedforward Neural Networks

As mentioned in the Introduction, the learning process of any machine learning model consists in adjusting the model's parameters to minimize a loss function, which is chosen according to the task at hand and in such a way that it is able to measure the discrepancy between the model's predictions and the actual targets in the data. In the context of feedforward neural networks, training consists in adjusting the weights and biases of each of the neurons in the network in order to minimize the loss function. This training process involves two steps: the feedforward process, in which the output of the network given an input vector is computed, and the backpropagation algorithm, in which the gradient of the loss function with respect to the weights and biases of the network is computed and these weights and biases are updated following the algorithm of gradient descent. In this section we formalize and explain all the concepts involved in training feedforward neural networks in detail.

Loss Function

A **loss function**, also known as a cost or objective function, is a function that measures the discrepancy between the predicted output $\hat{y}$ (produced by the network) and the true output y, which is the target label for the input data.
Let us proceed with a formal definition for loss function.

Definition 2.4. *[3] A **loss function** ℓ is defined as a mapping of $\hat{y}^{(i)}$ with its corresponding $y^{(i)}$ to a real number $l \in \mathbb{R}$, which captures the similarity between $\hat{y}^{(i)}$ and $y^{(i)}$. Aggregating over all the points in the dataset we find the overall loss, $\mathcal{L}$:*

$$\mathcal{L}(W, b) = \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{y}^{(i)}, y^{(i)}).$$

Here,

- N is the number of points in the dataset,
- W and b represent the collection of all weights and biases in the network,
- $\hat{y}^{(i)}$ is the predicted output vector for the i-th training example,
- $y^{(i)}$ is the true output vector for the i-th training example.

Remark 2.5. Although in practice this is not the case, any loss function ℓ treated in this project will be differentiable, since the optimization algorithm which we explain later for minimizing $\mathcal{L}$, requires the computation of the gradient of $\mathcal{L}$,

which is the average over the number of training examples N of the individual gradients of ℓ for each training example.
In practice, there is a list of mathematical properties that a loss function may or may not satisfy, and according to these properties, the optimization method chosen for computing the gradient of the loss function in order to minimize it may vary. [3] provides a fully detailed explanation of these properties and the implications of each of them.

Here are some examples of loss functions widely used in practice.

Example 2.6. Cross-Entropy Loss commonly used for classification tasks:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m} y_j^{(i)} \log(\hat{y}_j^{(i)}),$$

where $y_j^{(i)}$ is the true label (0 or 1) for the j-th class of the i-th training example, and $\hat{y}_j^{(i)}$ is the predicted probability for the j-th class of the i-th training example.
It is suitable for both binary and multi-class classification tasks. By minimizing the cross-entropy loss, we are minimizing the difference between the predicted probabilities for each class and the true labels.
For instance, in an m-class classification task, if the true label for an arbitrary example is the k-th class, with $1 \leq k \leq m$, namely $y = (0, \ldots, 1, \ldots, 0)$ where the k-th entry is 1 and the rest are 0, and the vector of predicted probabilities for each class (usually produced by the softmax function) is $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_k, \ldots, \hat{y}_m)$, then the cross-entropy loss for this example is $-\log(\hat{y}_k)$, which is minimized when $\hat{y}_k$ is 1 and the rest of the entries are 0.

Example 2.7. Mean Squared Error (MSE) commonly used for regression tasks:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2.$$

When minimizing the MSE, we are minimizing the average squared difference between the predicted output and the true output, which is a common goal in regression tasks as it leads to the model predicting the true output as closely as possible.

Recapping, the loss function allows us to evaluate how well the network is performing on predicting the target variable for the data examples, and thus by optimizing the loss function, we would ideally be able to improve the network's performance. In order to optimize the loss function, we want to know how the loss function changes as we change the weights and biases of the network.

Feed Forward Process

Until now, we have mentioned outputs of neurons, layers and even neural networks themselves, although we have only formalized how the output for an artificial neuron is produced. Let us now formalize the processes that take place in a feedforward neural network when computing the output. The process of computing the output of a neural network given an input vector involves computing the output of each layer inside the network, and it is called the **feed forward process**. Suppose we have a neural network with L layers. For each layer l with $1 \leq l \leq L$, let:

- $W^{(l)}$ be the weight matrix associated with layer l, that is, the matrix whose rows are the weight vectors of the neurons in layer l,
- $b^{(l)}$ be the bias vector associated with layer l, that is, the column vector whose entries are the biases of the neurons in layer l,
- $\phi^{(l)}$ be the activation function in layer l, that is, the activation function used in the neurons in layer l.

Now, if layer l has k neurons and layer $l-1$ has j neurons, then the output of layer l is given by

$$a^{(l)} = \begin{bmatrix} a_1^{(l)} \\ a_2^{(l)} \\ \vdots \\ a_k^{(l)} \end{bmatrix} = \phi^{(l)}\left(z^{(l)}\right), \tag{2.1}$$

where the expression $\phi^{(l)}(z^{(l)})$ indicates the element-wise application of the activation function $\phi^{(l)}$ to the vector $z^{(l)}$ and

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)} = \begin{bmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots & w_{1j}^{(l)} \\ w_{21}^{(l)} & w_{22}^{(l)} & \cdots & w_{2j}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1}^{(l)} & w_{k2}^{(l)} & \cdots & w_{kj}^{(l)} \end{bmatrix} \begin{bmatrix} a_1^{(l-1)} \\ a_2^{(l-1)} \\ \vdots \\ a_j^{(l-1)} \end{bmatrix} + \begin{bmatrix} b_1^{(l)} \\ b_2^{(l)} \\ \vdots \\ b_k^{(l)} \end{bmatrix}. \tag{2.2}$$

$a^{(l)}$ serves as the input to layer $l+1$. If $x \in \mathbb{R}^n$ is the input vector for the neural network, then the input of the first layer is $a^{(0)} = x$ and the output of the last layer and thus of the neural network as a whole is $a^{(L)}$.

Gradient Descent

Recalling, the goal of training a neural network is to minimize a function, the loss function, by adjusting its parameters, the weights and biases. Let us explain what is the gradient descent algorithm and how it is used to minimize an arbitrary function before explaining how it is used to minimize the loss function of a neural network. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function of n variables $x_1, x_2, \ldots, x_n$. Then the change Δf in f produced by a small change $\Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)^T$ in the variables is given by

$$\Delta f \approx \nabla f \cdot \Delta x, \tag{2.3}$$

where ∇f is the gradient of f at the point x, formally defined below.

Definition 2.8. *We define the **gradient** of a scalar-valued differentiable function f of several variables $x_1, x_2, \ldots, x_n$ at a point x as the vector of partial derivatives of f at x, expressed as:*

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)^T.$$

If the gradient of a function is non-zero at a point x, the direction of the gradient is the direction in which the function increases most quickly from x, or informally, the direction of steepest ascent of the function. As a consequence, the negative of the gradient is the direction of steepest descent of the function. Following this idea, we can choose

$$\Delta x = -\alpha \nabla f(x), \tag{2.4}$$

for a small, positive parameter $\alpha \in \mathbb{R}$, known as the **learning rate**. Then, equation 2.3 becomes

$$\Delta f \approx -\alpha \nabla f(x) \cdot \nabla f(x) = -\alpha \|\nabla f(x)\|^2,$$

where the notation $\|\cdot\|$ denotes the Euclidean norm of a vector. Since $\|\nabla f(x)\|^2 \geq 0$, this implies that $\Delta f \leq 0$, which means that the function will decrease (within the limits of the approximation in Equation 2.3), as long as we change x according to the following update rule derived from Equation 2.4:

$$x_{\text{new}} = x_{\text{old}} - \alpha \nabla f(x_{\text{old}}). \tag{2.5}$$

We can think of this update rule as defining the gradient descent algorithm [1]. In order to allow Gradient Descent to work as intended, we must choose a small enough learning rate α, so that the approximation in Equation 2.3 is accurate enough. If the learning rate is too large, the approximation may not hold and the function may not decrease, or it may even increase. On the other hand, if

the learning rate is too small, the algorithm may take too long to converge to the minimum of the function. In practice, the learning rate is often varied during the training process, decreasing as the algorithm approaches the minimum of the function, which is known as **learning rate scheduling**.

Remark 2.9. Gradient Descent was introduced by Agoustin-Louis Cauchy in 1847. In 1907, Jacques Hadamard independently proposed a similar method [7].

If we were to apply the gradient descent algorithm to minimize a loss function of the form $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \ell_i$, where ℓ_i is the loss function for the i-th training example, we would have to compute the gradients $\nabla \ell_i$ for each training example separately, and then average them to obtain the gradient $\nabla \mathcal{L}$ of the loss function. This is computationally expensive, since the number of training examples N can be very large. To avoid this, we can use the **stochastic gradient descent** algorithm [2], which approximates the gradient $\nabla \mathcal{L}$ by computing $\nabla \ell_i$ for a small randomly sampled subset of the training examples, called a **mini-batch**. The number of samples contained in each mini-batch is called **batch size**.

In practice, the gradient descent algorithm is often modified to improve its performance, giving arise to new **optimization algorithms**. However, to limit the scope of this work, we will not discuss these in detail and we will just mention some of the most popular along with references for the reader to further explore the details on these algorithms.

- **Adagrad** [4]: adapts the learning rate of each parameter based on the historical gradients of that parameter.
- **RMSprop** [5]: a modification of Adagrad which divides the learning rate by an exponentially decaying average of the squared gradients.
- **Adam** [6]: combines the benefits of Adagrad and RMSprop. Particularly useful for large datasets and high dimensional parameter spaces.

Backpropagation

Now that we know that in order to train a neural network we need to minimize a loss function, which we choose to be of a certain form such that it can be minimized using the gradient descent algorithm, we need to explain how to compute the gradient of the loss function in order to update the weights and biases of the network.

Backpropagation is the algorithm used to compute the gradient of the loss function with respect to the weights and biases of the network and update these using an optimization algorithm, in this case, gradient descent.

Backpropagation is carried out for each training example in the following steps: First, perform a feedforward pass to compute the output $a^{(L)}$ for the given training example. Now, we want to compute the gradient of the loss function

$$\nabla \ell = \left(\dots, \frac{\partial \ell}{\partial w_{jk}^{(l)}}, \dots, \frac{\partial \ell}{\partial b_j^{(l)}}, \dots \right),$$

where $w_{jk}^{(l)}$ denotes the weight connecting the j-th neuron in layer l to the k-th neuron in layer $l-1$ and $b_j^{(l)}$ denotes the bias term for the j-th neuron in layer l. Using the chain rule, we can express the gradient of the loss function with respect to the weights and biases of the network as:

$$\frac{\partial \ell}{\partial w_{jk}^{(l)}} = \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial \ell}{\partial a_j^{(l)}} = a_k^{(l-1)} \phi'^{(l)}(z_j^{(l)}) \frac{\partial \ell}{\partial a_j^{(l)}},$$

$$\frac{\partial \ell}{\partial b_j^{(l)}} = \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \frac{\partial \ell}{\partial a_j^{(l)}} = \phi'^{(l)}(z_j^{(l)}) \frac{\partial \ell}{\partial a_j^{(l)}},$$

where $z_j^{(l)}$ is the j-th component of the vector $z^{(l)}$ given in equation 2.2, $a_j^{(l)}$ is the output of the j-th neuron in layer l and $\phi'^{(l)}$ is the derivative of the activation function used in layer l. If layer l is the output layer, namely when $l = L$, we can compute $\frac{\partial \ell}{\partial a_j^{(L)}}$ using the predicted and true outputs ($a^{(L)}$ and y) for our training example. Otherwise,

$$\frac{\partial \ell}{\partial a_j^{(l)}} = \sum_{i=1}^{n_{l+1}} \frac{\partial z_i^{(l+1)}}{\partial a_j^{(l)}} \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}} \frac{\partial \ell}{\partial a_i^{(l+1)}},$$

where n_{l+1} is the number of neurons in layer $l+1$.
Backpropagation starts from the output layer L and moves backwards through each layer l to the input layer. Finally, the weights and biases of the network are updated using the chosen optimization algorithm's update rule. Using gradient descent, we would update the weights and biases as follows:

$$w_{jk}^{(l)} = w_{jk}^{(l)} - \alpha \frac{\partial \ell}{\partial w_{jk}^{(l)}},$$

$$b_j^{(l)} = b_j^{(l)} - \alpha \frac{\partial \ell}{\partial b_j^{(l)}},$$

where α is the learning rate. Now, we can repeat the process for each training example in the dataset, and iterate over the entire dataset multiple times until the loss function converges to a minimum.

Remark 2.10. If we chose to use Stochastic Gradient Descent for optimization, supposing w_k and b_l denote the weights and biases of the network, we would pick a randomly sampled mini-batch of m training examples and we would update the weights and biases as follows:

$$w_k \to w_k' = w_k - \alpha \sum_{j=1}^{m} \frac{\partial \ell_j}{\partial w_k},$$

$$b_l \to b_l' = b_l - \alpha \sum_{j=1}^{m} \frac{\partial \ell_j}{\partial b_l},$$

where ℓ_j is the loss function for the j-th training example in the current mini-batch. Then we would pick out another randomly chosen mini-batch and train with those samples. And so on, until we we have exhausted the training inputs, which is said to complete an **epoch** of training. We can repeat this process for multiple epochs until the loss function converges to a minimum.

Related concepts

Before concluding this chapter, we introduce some related concepts which are relevant in the context of training neural networks and that we will use in the following chapters.

Overfitting

Overfitting is a phenomenon in machine learning where a model learns the training data too well during training, to the point where it performs poorly on unseen data, for instance the validation set. A cause for overfitting could be carrying on training for too many iterations, making the parameters of the model adjust to fit the noise in the training data or remembering specific examples in the training data, rather than capturing general underlying patterns [9]. Overfitting can be detected by comparing the performance of the model on the training set and the validation set. We know a model is starting to overfit when the performance on the validation set starts to degrade while the performance on the training keeps improving. In deep learning, some common techniques for preventing overfitting are:

- **Early Stopping:** Stop training the model when the performance on a validation set starts to degrade.
- **Dropout:** Randomly set a fraction of the neurons' outputs to zero during each training iteration.

- **Data Augmentation:** Increase the amount of examples in the training dataset by applying transformations to the data.

Embeddings

As a general concept, embeddings can be defined as follows:

Definition 2.11. *Given an input $x \in \mathbb{R}^n$, an **embedding** corresponding to x is a vector $e \in \mathbb{R}^d$ given by a mapping function*

$$E : \mathbb{R}^n \to \mathbb{R}^d$$

where d is the dimension of the embedding space.

In the context of neural networks, given an input vector x, the corresponding embedding vector e is the output of an arbitrarily (although not randomly) chosen layer in the network during the feedforward process for x, we call this layer the **embedding layer**. The embedding layer maps input information from a high-dimensional to a lower-dimensional space, allowing the network to learn more about the relationship between inputs and to process the data more efficiently [10]. The dimension of the embedding space corresponds to the number of neurons in the embedding layer or its dimension.

Evaluation Metrics

An **evaluation metric** is a measure used to evaluate the performance of a machine learning model. Each task has its own set of evaluation metrics, which are chosen based on the desired outcome of the model.
Some common evaluation metrics for classification tasks are:

- **Accuracy:** The proportion of correctly classified examples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

- **Precision:** The proportion of true positive predictions among all positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}},$$

  where True Positives is the number of correctly predicted positive examples and False Positives is the number of negative examples incorrectly predicted as positive.

- **Recall:** The proportion of true positive predictions among all actual positive examples.
$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

Remark 2.12. Note that as defined above, precision and recall are only suitable for binary classification tasks. For multi-class classification tasks, these metrics are extended to the multi-class setting.

Some common evaluation metrics for regression tasks are:

- **Mean Squared Error (MSE):** As defined above when talking about loss functions, MSE is the average of the squared differences between the predicted and true values.
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2$$

- **Mean Absolute Error (MAE):** The average of the absolute differences between the predicted and true values.
$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}^{(i)} - y^{(i)}|$$

Chapter 3

Diversity in Deep Learning

In this chapter, we briefly expand the importance of diversity in deep learning models, we introduce some metrics used to evaluate diversity in machine learning, with a focus on the Vendi Score, a metric derived from ecology, which we will use in our experiments in Chapter 4 and which has numerous advantages over other metrics, reviewed in this chapter. Before introducing the Vendi Score, we will dedicate some space to explain how diversity is treated in ecology, which metrics are used to measure it, which properties must these metrics satisfy and what are their limitations, focusing on the Hill numbers, which are the basis for the Vendi Score. We then introduce the Vendi Score and explain its properties through the required definitions and lemmas. We end by extending the Vendi Score to a family of Vendi Scores which extends the applicability of this metric to problems involving datasets with imbalanced classes, among others.

We define **diversity** as the variety of examples that a model can capture in the data. The following points summarize the importance of diversity in deep learning models:

- **Generalization:** A diverse model is more likely to generalize well to unseen data, as it can capture a broader range of patterns in the data.
- **Robustness:** Diverse models are more robust to noise and perturbations in the data, as they are less likely to overfit to specific features.
- **Creativity:** In generative deep learning, diversity is crucial for generating novel and varied outputs. For instance, in molecular generative modeling, diversity enables the discovery of new molecules with potential applications in medicine and materials science. Evaluating diversity also helps prevent issues like mode collapse, where a model only generates a limited range of outputs.

Diversity Metrics in Machine Learning

In order to evaluate diversity, researchers across many machine learning fields have developed or even adapted existing metrics. In this section, we will introduce some of the most used metrics along with their shortcomings, comparing them to the Vendi Score, a similarity-based metric which we will introduce in a subsequent section.

[21] Machine Learning researchers often use some form of average pairwise similarity to quantify diversity, e.g. pairwise-BLEU [24] and D-Lex Sim [25] for text data or IntDiv for molecular data [26]. However, as discussed in [20], average similarity can fail to effectively capture diversity, even in simple scenarios, for instance it can score two populations with the same number of components/species but different levels of per-component variance the same. This is not the case for the Vendi Score, which accounts for correlations between features and is able to capture the increased diversity resulting from combining distinct similarity functions.

Other metrics used to evaluate diversity, especially in computer vision, include a two-metric evaluation paradigm using precision and recall [27] and Fréchet Inception distance (*FID*) [28], which measures the Wasserstein-2 distance between two Gaussian distributions, one Gaussian fit to the embeddings of the reference sample and another one fit to the embeddings of the sample to be evaluated for diversity. However, these metrics are less flexible as they rely on a reference distribution, and in the case of *FID*, additionally require the availability of an embedding function. Compared to these approaches, the Vendi Score can measure diversity without relying on a reference distribution or dataset and is therefore more general.

Some other existing metrics evaluate diversity using a pre-trained classifier, therefore requiring labeled datasets. For example, the Inception score (*IS*) [29], which evaluates diversity using the entropy of the marginal distribution of class labels predicted by an ImageNet classifier. Another example is number of modes (*NOM*) [30], a metric used to evaluate the diversity of *GAN*s (Generative Adversarial Networks). *NOM* is calculated by using a classifier trained on a labeled dataset and then counting the number of unique labels predicted by the classifier when using samples from a *GAN* as input. Both *IS* and *NOM* define diversity in terms of predefined labels, and therefore require knowledge of the ground truth labels and a separate classifier, which limits their applicability. Once again, the Vendi Score does not suffer from these limitations, as it can be used to evaluate diversity without relying on labeled data or a pre-trained classifier.

Recapping, several attempts have been made to measure diversity in machine learning. However, the proposed metrics can be limited in their applicability as they require a reference dataset or predefined labels, or are domain-specific and

applicable to one class of models. Besides, these metrics do not give their user the ability to decide which type of diversity they want to measure. The existing metrics that do not have those applicability limitations have serious shortcomings when it comes to capturing diversity. These shortcomings are addressed by the Vendi Score, as we will see in the upcoming sections.

3.1 Ecological Diversity and Hill Numbers

Ecologists have been studying diversity of species in ecosystems for decades, and they have developed a variety of metrics that capture intuitive aspects of diversity. Some of the most used metrics in ecology are the Hill numbers [17] and the triplets alpha diversity, beta diversity, and gamma diversity [18]. However, these metrics have some drawbacks.
The Vendi Score is a new metric based on the Hill numbers that aims to alleviate many of the challenges faced by the commonly used diversity metrics in ecology and machine learning.
In this section, we introduce ecological diversity and the Hill numbers in detail, along with their properties and limitations.

Ecological Diversity

A **diversity index** is a measure that quantifies the diversity of a population of species by reflecting how many different types (e.g., species) there are. Consider a probability distribution $p = (p_1, \ldots, p_S)$ on a space $X = \{1, \ldots, S\}$. Ecologists refer to each member of X as a species and to the individual probability p_i as the relative abundance of the i-th species in X. Most ecological diversity indices do not account for species similarity, assuming all species in a community are completely dissimilar and defining diversity only in terms of the relative abundance p. Ecologists have proposed a number of properties which encode basic scientific intuition, stating that any diversity measure taking species similarity into account should satisfy [12]. These properties apply to a sequence $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ of functions, that is, a diversity measure for communities modelled as finite probability distributions. Here, the similarities between species are encoded in a $S \times S$ matrix $Z = (Z_{ij})$, with Z_{ij} representing the similarity between species i and j. We assume that $0 \leq Z_{ij} \leq 1$, with 0 indicating complete dissimilarity and 1 indicating complete similarity (identical) species. Hence we also have $Z_{ii} = 1$. For $S \geq 1$, we write

$$\Delta_S = \{\text{probability distributions on } \{1, \ldots, S\}\},$$

and

$$u_S = \left(\frac{1}{S}, \ldots, \frac{1}{S}\right),$$

which denotes the uniform distribution on n elements or species.
Geometrically, Δ_S is the standard $(S-1)$-dimensional simplex and u_S is the center of this simplex.

1. **Effective number**
   Let $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ be a diversity measure. Then D^Z is an **effective number** if $D^Z(u_S) = S$ for all $S \geq 1$.
   In other words, a population containing S equally abundant, completely dissimilar species should have a diversity score of S. In the other hand, if all species are identical, namely, if one species accounts for the total of the community, the diversity should be minimized and equal to 1.

2. **Modularity**
   Suppose the community X is partitioned into m disjoint subcommunities and species within subcommunities are totally dissimilar. Then the diversity of the whole community should be entirely determined by the sizes and diversities of the subcommunities.
   Modularity enables us to calculate the diversity of a partitioned community from the diversities and sizes of the subcommunities alone, without having to know the abundance and similarity data within the subcommunities. [12] gives the formula for the diversity of a partitioned community in terms of the diversities and sizes of the subcommunities with its corresponding proof.

3. **Replication principle**
   Following the context of the previous property, if these m subcommunities within the system have the same relative abundance distribution p (meaning they are of equal size and diversity $D^Z(p)$), then the diversity of the whole system X should be $mD^Z(p)$.
   Formally, a diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ satisfies the **replication principle** if for all $m, S \geq 1$ and $p \in \Delta_S$, we have:

   $$D^Z(u_m \otimes p) = mD^Z(p),$$

   where $\otimes$ denotes the tensor product [13] of probability distributions.
   When $m = 2$, replication is called "doubling". This concept is used and treated in [17], for instance.

4. **Symmetry**
   A diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ is said to be **symmetric** if

   $$D^Z(p) = D^Z(p_{\sigma(1)}, \ldots, p_{\sigma(S)}),$$

for all $p \in \Delta_S$ and σ a permutation of $\{1, \ldots, S\}$.
This means that the diversity of a population should not depend on the order in which the species are listed.

5. **Absence-invariant**
A diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ is said to be **absence-invariant** if whenever $p \in \Delta_S$ and $1 \leq i \leq S$ with $p_i = 0$, then:

$$D^Z(p) = D^Z(p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_S).$$

6. **Identical species**
If two species are identical, then merging them into a single species should not change the diversity of the community.
Formally, let Z be a $S \times S$ similarity such that $Z_{i,S} = Z_{i,S-1}$ and $Z_{S,i} = Z_{S-1,i}$ for all i. Let $p \in \Delta_S$. Write Z' for the restriction of Z to the first $S-1$ species, and define $p' \in \Delta_{S-1}$ by

$$p'_j = \begin{cases} p_j & \text{if } j < S-1 \\ p_{S-1} + p_S & \text{if } j = S-1 \end{cases}.$$

Then a diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ verifies the **identical species** property if $D^{Z'}(p') = D^Z(p)$.

7. **Monotonicity**
When the similarities between species are increased, diversity decreases.
Formally, let Z and Z' be $S \times S$ matrices such with $Z_{ij} \leq Z'_{ij}$ for all i, j. Then a diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ is said to be **monotone** if $D^Z(p) \geq D^{Z'}(p)$, for all $p \in \Delta_S$.

8. **Naive Model**
When similarities between species are ignored, diversity is greater than when they are taken into account. This property is an extreme case of monotonicity, if a measure knows nothing of the commonalities between species, it will evaluate the community as more diverse than it really is.

9. **Range**
The diversity of a community of S species is between 1 and S.
Formally, a diversity measure $(D^Z : \Delta_S \to (0, \infty))_{S \geq 1}$ is said to verify the **range** property if $D^Z(p) \in [1, S]$ for all $p \in \Delta_S$ and $S \geq 1$.

In the next section we define the Hill numbers, and we will show that they do not consider similarity among species, which makes them not suitable for satisfying some of the above properties. In the subsequent section, we will introduce

and show that the Vendi Score takes into account the similarity between species and satisfies all the properties described above.

Hill numbers

Before defining the Hill numbers, we may introduce the concept of generalized means, since it will be useful to understand the Hill numbers.

Definition 3.1. *[31] If p is a non-zero real number, given a set of n positive real numbers $x_1, x_2, \ldots, x_n$, the **generalized mean** of order p is defined as:*

$$M_p(x_1, x_2, \ldots, x_n) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{1/p}.$$

Example 3.2. The **arithmetic mean** is the generalized mean of order 1:

$$M_1(x_1, x_2, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Definition 3.3. *[16] Consider a probability distribution $p = (p_1, \ldots, p_S)$ on a space $X = \{1, \ldots, S\}$. We define the **Hill number of order** q as the reciprocal of the weighted generalized mean M_{q-1} of the proportional abundances of the species in the population:*

$${}^qD = \frac{1}{M_{q-1}} = \left(\sum_{i=1}^{S} p_i^q \right)^{1/(1-q)}, q \geq 0, q \neq 1.$$

Where $p = (p_1, \ldots, p_S)$ is the relative abundance vector of the species in the population and S is the species richness (the number of different types within the dataset). All Hill numbers are in units of "species".

With respect to q:

- When $q = 0$, the species abundances do not count at all and the species richness ${}^0D = S$ is obtained.
- When $q = 1$, the metric is undefined, but its limit as q approaches 1 is well-defined and gives:

  $${}^1D = \lim_{q \to 1} {}^qD = \exp\left(-\sum_{i=1}^{S} p_i \log(p_i) \right) = \exp(-H(p)),$$

  where $H(p)$ is the Shannon entropy of the distribution p. In this case, the species are weighed in proportion to their frequencies, and the measure ${}^1D$ can be interpreted as the effective number of common species (species with typical abundances) in the community.

- When $q = 2$, abundant species are favored and rare species are discounted. The measure ${}^2D$ can be interpreted as the effective number of dominant or very abundant species in the community.

- When $q = \infty$, all the weight is given to the most abundant species, and the Hill number is also a limit:

$$ {}^{\infty}D = \lim_{q \to \infty} {}^{q}D = \exp\left(-\log\left(\max_i p_i\right)\right) = \frac{1}{\max_i p_i}. $$

In practice, q acts as a sensitivity parameter, controlling the relative emphasis that the user wishes to place on common and rare species.
As we said before and are able to deduce from the definition, Hill numbers do not take into account the similarity between species and thus do not satisfy all the properties of an ideal diversity measure. Nevertheless, Chapter 7 of [14] shows, using results on generalized means from [15], the list of properties satisfied by the Hill numbers.

Recapping, Hill numbers, although interpretable and grounded in intuitive notions of diversity, have important drawbacks which limit their applicability:

1. They assume some concept of classes and an ability to classify samples within classes.

2. They assume knowledge of an abundance vector p quantifying the number of elements in each class.

3. They do not take into account similarity between species, therefore not verifying all the properties of an ideal diversity measure.

In order to lift these limitations of the Hill numbers, the Vendi Score was introduced in [20], providing a solution for $q = 1$. Later, in [21], the Vendi Score was extended to the other Hill numbers, providing a solution for all q.

3.2 Vendi Score

The Vendi Score is born from the idea of extending the way of thinking about diversity in ecology to machine learning. The Vendi Score is designed to be a diversity metric that only relies on the samples being evaluated for diversity, which achieves its maximum value when all samples are dissimilar and its minimum value when all samples are the same, and achieves this by using a similarity function or kernel on the samples.
Before defining the Vendi Score, let us define similarity function or kernel.

Definition 3.4. *[19] We define **similarity function** or **kernel** in a vector (feature) space X as an inner-product, that is, a function $k : X \times X \to \mathbb{R}$ that satisfies the following properties:*

- $k(x, x) \geq 0$ *for all* $x \in X$ *and* $k(x, x) = 0$ *if and only if* $x = 0$ *(positive-definite),*
- $k(x, y) = k(y, x)$ *for all* $x, y \in X$ *(symmetry),*
- $k(\alpha x, y) = \alpha k(x, y)$ *for all* $x, y \in X$ *and* $\alpha \in \mathbb{R}$*,*
- $k(x + y, z) = k(x, z) + k(y, z)$ *for all* $x, y, z \in X$*.*

Definition 3.5. *(Vendi Score). Let $x_1, \ldots, x_n \in X$ denote a collection of samples, let $k : X \times X \to \mathbb{R}$ be a positive semidefinite similarity function, with $k(x, x) = 1$ for all x, and let $K \in \mathbb{R}^{n \times n}$ denote the kernel matrix with entry $K_{i,j} = k(x_i, x_j)$. Denote by $\lambda_1, \ldots, \lambda_n$ the eigenvalues of K/n. The Vendi Score (VS) is defined as the exponential of the Shannon entropy of the eigenvalues of K/n:*

$$VS_k(x_1, \ldots, x_n) = \exp\left(-\sum_{i=1}^{n} \lambda_i \log(\lambda_i)\right), \tag{3.1}$$

where we use the convention $0\ log(0) = 0$.

Remark 3.6. Notice that in Definition 3.5, the similarity function k must verify $k(x, x) = 1$ for all x, while in the general definition of a similarity function, this is not a requirement. Following these requirements, the similarity function k is user-defined in practice.

The proofs corresponding to the following lemmas and theorems can be found in the appendix of [20].
The following Lemma proves that the Vendi Score is well-defined.

Lemma 3.7. *Let $K \in \mathbb{R}^{n \times n}$ denote a positive semi-definite kernel matrix with $K_{i,i} = 1$ for all i. Denote by Δ_n the $(n-1)$-dimensional simplex. Let $\lambda = (\lambda_1, \ldots, \lambda_n)$ denote the vector of eigenvalues of K/n. Then $\lambda \in \Delta_n$.*

Lemma 3.7 implies that, in particular, the eigenvalues of K/n as defined in Definition 3.5 are nonnegative and sum to 1. The Shannon entropy of the eigenvalues of K/n is thus well-defined and the Vendi Score is well-defined.
By the following lemma, the Vendi Score can be expressed directly as a function of the kernel similarity matrix K. This expression is useful for proving upcoming lemmas and theorems.

Lemma 3.8. *Consider the same setting as Definition 3.5. Then*

$$VS_k(x_1, \ldots, x_n) = \exp\left(-tr\left(\frac{K}{n} log \frac{K}{n}\right)\right).$$

The formulation of the Vendi Score given by Definition 3.5 assumes that the collection of samples $x_1, \ldots, x_n$ were sampled independently. However, this metric can be extended to the case where there is an explicit probability distribution over the sample space, given by the following definition.

Definition 3.9. *(Probability-Weighted Vendi Score). Let $p \in \Delta_n$ denote a probability distribution on a discrete space $X = \{x_1, \ldots, x_n\}$. Let:*

- *$k : X \times X \to \mathbb{R}$ be a positive semidefinite similarity function, with $k(x, x) = 1$ for all x.*
- *$K \in \mathbb{R}^{n \times n}$ denote the kernel matrix with $K_{i,j} = k(x_i, x_j)$.*
- *$\tilde{K}_p = diag(\sqrt{p})K\, diag(\sqrt{p})$ denote the probability-weighted kernel matrix.*
- *$\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of $\tilde{K}_p$.*

The Vendi Score (VS) is defined as the exponential of the Shannon entropy of the eigenvalues of $\tilde{K}_p$:

$$VS_k(x_1, \ldots, x_n, p) = \exp\left(-\sum_{i=1}^{n} \lambda_i \log(\lambda_i)\right).$$

Remark 3.10. Equation 3.1 is a special case of the probability-weighted Vendi Score where the probability distribution is uniform (i.e., $p = (1/n, \ldots, 1/n)$).

Lemma 3.11. *Let $p \in \Delta_n$ be a probability distribution over $\{x_1, \ldots, x_n\}$ and suppose $k(x_i, x_j) = 0$ for all $i \neq j$. Then the Vendi Score is the exponential of the Shannon entropy of p,*

$$VS_k(x_1, \ldots, x_n, p) = \exp(-H(p)).$$

The following theorem summarizes some of the properties possessed by the Vendi Score which are part of the stated in Section 3.1.

Theorem 3.12. *(Properties of the Vendi Score). Consider the same definitions in Definition 3.5 and Definition 3.9.*

1. ***Effective number**. If $k(x_i, x_j) = 0$ for all $i \neq j$, then $VS_k(x_1, \ldots, x_n)$ is maximized and equal to n. If $k(x_i, x_j) = 1$ for all i, j, then $VS_k(x_1, \ldots, x_n)$ is minimized and equal to 1.*

2. ***Identical elements***. *Suppose* $k(x_i, x_j) = 1$ *for some* $i \neq j$. *Let* p' *denote the probability distribution created by combining i and j, i.e.,* $p'_i = p_i + p_j$ *and* $p'_j = 0$. *Then the Vendi Score is unchanged,*

$$VS_k(x_1, \ldots, x_n, p) = VS_k(x_1, \ldots, x_n, p').$$

3. ***Partitioning***. *Suppose* $S_1, \ldots, S_m$ *are collections of samples such that, for any* $i \neq j$, *for all* $x \in S_i$, $x' \in S_j$, $k(x, x') = 0$. *Then the diversity of the combined samples depends only on the diversities of* $S_1, \ldots, S_m$ *and their relative sizes. In particular, if* $p_i = |S_i| / \sum_j |S_j|$ *is the relative size of* S_i *and* $H(p_1, \ldots, p_m)$ *denotes the Shannon entropy, then the Vendi Score is the geometric mean:*

$$VS_k(S_1, \ldots, S_m) = \exp(H(p_1, \ldots, p_m)) \prod_{i=1}^{m} VS_k(S_i)^{p_i}.$$

4. ***Symmetry***. *If* $\pi_1, \ldots, \pi_n$ *is a permutation of* $1, \ldots, n$, *then*

$$VS_k(x_1, \ldots, x_n) = VS_k(x_{\pi_1}, \ldots, x_{\pi_n}).$$

The rest of the properties which are not proven to be verified by the Vendi Score in Theorem 3.12 can be easily proven either by being derived from these proven properties or by using the lemmas and definitions previously presented in this section.

Contrary to many diversity metrics in ecology, the Vendi Score accounts for similarity and does not require knowledge of the prevalence of the categories in the collection to be evaluated for diversity. However, the Vendi Score treats each item in a given collection with a level of sensitivity proportional to the item's prevalence. This is undesirable in settings where there is a significant imbalance in item prevalence. In the following section Cousins of the Vendi Score, we introduce a family of diversity metrics—Vendi scores with different levels of sensitivity controlled by the order parameter q—which can be used in a variety of applications [21].

3.2.1 Cousins of the Vendi Score

As we mentioned earlier, the Vendi Score introduced by [20] which we reviewed in the previous section only provides an extension for the Hill number of order $q = 1$. And it was in [21] where the authors provided a theorem that relates the eigenvalues of a normalized similarity matrix to item prevalence and used this result to extend the Vendi Score to the other Hill numbers.
Let us present this theorem and the extended Vendi Scores.

Theorem 3.13. *[The Similarity-Eigenvalue-Prevalence Theorem] Let $(x_1, \ldots, x_N)$ denote a collection of elements, where each $x_i = (x_{i1}, \ldots, x_{iM_i})$ contains a unique element repeated M_i times, i.e., $x_{ij} = x_{ik}$ for all $j, k \in \{1, \ldots, M_i\}$. Define $C = \sum_{i=1}^{N} M_i$. Let $K \in \mathbb{R}^{C \times C}$ denote a kernel matrix such that $K(x_{i\cdot}, x_{j\cdot}) = 1$ when $i = j$ and 0 otherwise, $\forall i, j \in \{1, \ldots, N\}$. Denote by $\tilde{K} = \frac{K}{C}$ the normalized kernel. Then $\tilde{K}$ has exactly N non-zero eigenvalues $\lambda_1, \ldots, \lambda_N$ and $\lambda_i = \frac{M_i}{C}$ $\forall i \in \{1, \ldots, N\}$.*

Proof. The proof of this theorem can be found in the Appendix of [21]. □

The theorem states that under the assumption that members of different species are completely dissimilar there are as many nonzero eigenvalues of the species similarity matrix as there are species and that these eigenvalues are exactly equal to the prevalence of the different species. This theorem therefore provides a recipe for recovering the different Hill numbers exactly using the similarity-based construct the Vendi Score is based on. The benefit of computing the Hill numbers using the same similarity-based approach as the Vendi Score is it hints at the possibility of not needing to assume knowledge of species prevalence. Furthermore, the assumption of complete dissimilarity between species is strong and limits the Hill numbers' applicability. The similarity-based approach described above readily allows us to lift that assumption, by simply replacing the zero entries in the similarity matrix with a user-defined similarity function between species.
Endowed with Theorem 3.13, we can safely transition from Hill numbers—diversity indices that assume knowledge of species prevalence and that don't account for species similarity—to Vendi scores, diversity indices that do not assume knowledge of species prevalence and that effectively account for species similarity.

Definition 3.14. *We denote by $VS_q(x, k)$ the Vendi score of order q for the collection x under similarity function $k(\cdot, \cdot)$ and define,*

$$VS_q(x, k) = \exp\left(\frac{1}{1-q} \log \sum_{i \in supp(\lambda(x,k))} \lambda_i^q(x, k)\right),$$

where $\lambda(x, k)$ denotes the set of eigenvalues of the normalized similarity matrix induced by the input similarity function $k(\cdot, \cdot)$, and $supp(\lambda(x, k))$ denotes the indices for the nonzero eigenvalues.

Vendi scores are differentiable, which makes them suitable for gradient-based methods in machine learning and science. This differentiability enables us to go beyond simply evaluating diversity to effectively enforcing diversity, by embedding the scores into objective functions of interest [22].

Limitations of the Vendi Scores

Despite being a powerful tool for evaluating diversity, the Vendi Score still has some limitations.

- The choice of the similarity function to evaluate the Vendi Score does affect the behavior of the metric, the study of how the Vendi Score is modified by the choice of the kernel is a work in progress.
- The Vendi scores can be computationally expensive to compute in settings with large sets of data which do not have vector representations. Therefore, finding methods for scaling the scores when no embeddings are available, like large collections of raw textual data, remains an open problem.

Figure 2 shows the similarity matrices induced by a shape similarity function and/or a color similarity function. Each entry in the matrices shown is defined to be 1 if the column and the row share both shape and color, 0.5 if they share either shape or color, and 0 otherwise.

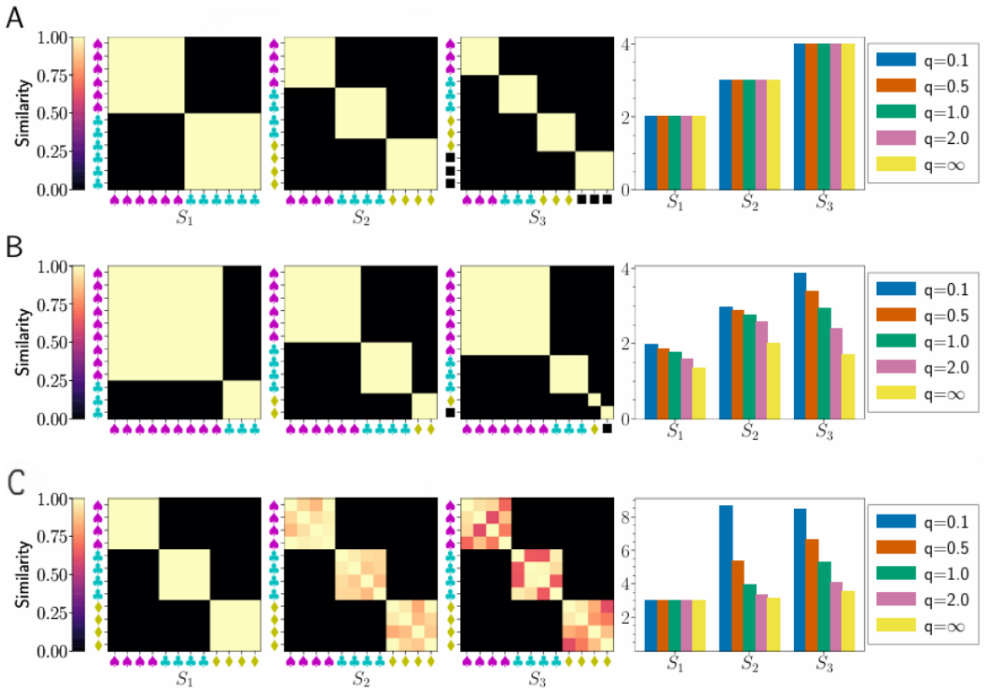

Figure 2: **Sensitivity of Different Vendi Scores Under Different Scenarios**. (Source: [21]) (A) Varying the number of classes under perfect balance. Each Vendi score measures the number of classes exactly; they are effective numbers. (B) Varying the number of classes under imbalance. Smaller orders q more accurately describe the correct number of modes. (C) $q = \infty$ gives a Vendi Score that is more resistant to intra-class variance. For smaller values of q, the Vendi scores increase with larger amounts of variance, although the Vendi score with $q = 0.1$ decreases slightly between example S_2 and S_3.

Chapter 4

Empirical Study

In this chapter, we present the experiments conducted using the metrics developed in the previous chapter. As explained in the introduction, our objective is to analyze the tradeoff between the performance of a model when carrying out the task it has been trained for and the performance of the same model when carrying out a different task for which it has not been thoroughly trained, and we want to do so by evaluating the diversity of the embeddings generated by the model.

We will start by presenting the setting of the experiments, in which we will describe the steps that make up the experiments, followed by our initial hypothesis for the results, namely what we think the behavior of the experiments will be and the reasons for this expected behavior. Finally we expose the results obtained and a discussion of these, comparing them to our hypothesis and extracting conclusions from them.

Setting

We constructed a toy model which allows us to simulate real world examples of applicability of our main objective but in a smaller scale and a reproducible manner. To minimize the margin of error of the experiments and ensure the reliability of the results, we carried out the procedure for the experiments using two different datasets separately. The chosen datasets were the MNIST and Fashion MNIST datasets. These were chosen because they are relatively small and thus computationally inexpensive to work with. Additionally, they are well-known and widely used in the machine learning community for benchmarking purposes and image classification tasks due to the previously exposed reasons.

In order to simulate the context of having a neural network trained for a task and wondering whether this network can be used to perform another task, we established a set of fine classes included into a set of metaclasses for each of the

datasets (for details on the specific classes and metaclasses in each dataset, refer to the section Description of the Used Datasets in the Appendix). and supposed our main task for the neural network in the toy model was classifying into the metaclasses of the dataset, while the secondary task was classifying into the fine classes of the dataset. We use a simple neural network with two 16-dimensional fully connected hidden layers. We chose the embedding layer of the neural network to be the second 16-dimensional fully connected hidden layer. Considering that in both datasets the number of fine classes is ten and the number of metaclasses is two, the dimension of the embedding layer of the network is big enough to prevent it from becoming a bottleneck for the diversity of the embeddings.

Procedure

For each dataset, the procedure is as follows: We start by training the network for the classification into the metaclasses of the dataset, implying that the network will have an output layer with dimension equal to the number of metaclasses in the dataset (hereafter, metaclass classifier output layer). For each training epoch, the accuracy on both the train and test sets is evaluated, and these values are stored. After each epoch of training of the network, we freeze the parameters (e.g. we make them fixed) of the hidden layers and we swap the metaclass classifier output layer by a new output layer suited to perform classification into the fine classes in the dataset (hereafter, fine class classifier ouput layer). Once the output layer has been swapped and the rest of the layers in the network have been frozen, the network is trained for a small number of epochs to classify into the fine classes of the dataset (notice that although we say that the network is trained, only the parameters of the fine class classifier output layer will actually change since the rest of parameters are fixed). The accuracy of this modified network for classifying into the fine classes is evaluated on the train and test sets and the resulting values are stored. These values allow us to know how the network is performing when classifying into the fine classes of the dataset. Before resuming the original training, we feed the neural network with the entire training dataset by batches in order to get the output of the embedding layer, namely the embeddings of the samples in the train dataset. Batch by batch, the Vendi Score of order 1 (hereafter, just Vendi Score. Order 1 was chosen because the three datasets used in the experiments contain the same number of samples per class, in other words, the classes are balanced in all datasets) is evaluated on the embeddings and the results are averaged and stored. The Vendi Score is the metric that allows us to know the diversity of the embeddings produced by the neural network in each training epoch while being trained for classifying into the metaclasses of the dataset. Finally, we swap the fine class classifier output layer back to the original metaclass classifier

output layer with its previous weights and we unfreeze all the previously frozen parameters. We resume the training of the original network until the end of the next epoch, when we repeat the process from the start.
The loss function used for both training processes is the categorical cross-entropy loss and the optimizer used is the Adam optimizer.
We use learning rate scheduler to adjust the learning rate during the training process and achieve higher accuracy scores. This scheduler can be set to monitor either the loss on the training set or the loss on the validation set of the metaclass classification task. Monitoring the loss on the training set can lead to overfitting the data, while monitoring the loss on the validation set can lead to a better generalization of the model. Hence we will replicate the procedure for both settings of the learning rate scheduler in order to get insights on how do our experiments change when overfitting the data and when favoring generalization. We will also replicate the procedure for different values of the batch size used in the training process to see how this parameter affects the results. The procedure ends when the objective number of training epochs has been reached.

4.1 Hypothesis and discussion of the results

In this section, we present the results obtained from the experiments and comment them in comparison to our initial hypothesis, which we present below. These results are presented in the form of plots containing the Vendi Score of the embeddings (*Vendi Score (order 1) of the embeddings*), the accuracy of the network when classifying into the metaclasses in both the train ((*Metaclass Classifier Accuracy*)) and test (*Metaclass Classifier Validation Accuracy*) sets and the accuracy of the network when classifying into the fine classes in both the train (*Fine classes Classifier Accuracy*) and test (*Fine classes Classifier Validation Accuracy*) sets over the training epochs.

The main hypothesis for the experiment is that the longer the neural network is trained for classifying the data into metaclasses and the more it overfits the training data, the more collapsed will the embeddings corresponding to samples of the same metaclass be, resulting in the Vendi Score of the embeddings stabilizing around the number of metaclasses of the dataset, and the accuracy of the network when classifying into the fine classes being lower, as the embeddings corresponding to samples of different fine classes but the same metaclass will be more likely to be collapsed into the same region of the embedding space, hence making it harder for the fine class classifier output layer to distinguish and classify these embeddings.

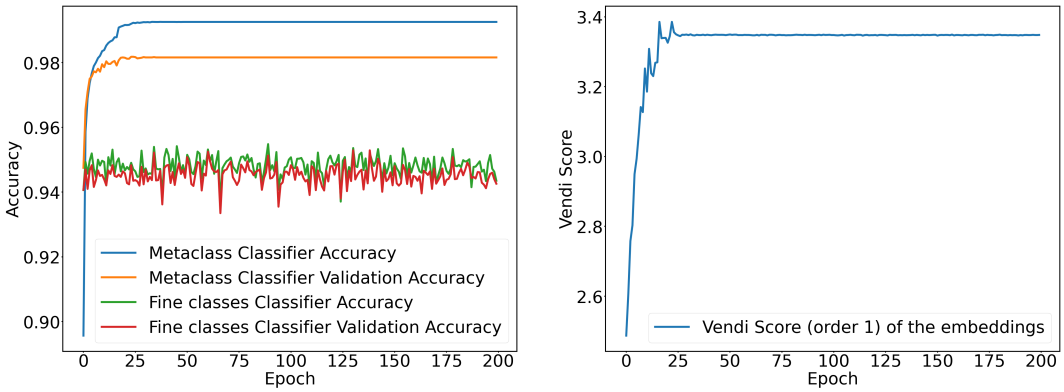

Figure 3: **Accuracy Scores and Vendi Score over training epochs on MNIST with batch size 256 and validation loss as the monitor for the learning rate scheduler**.

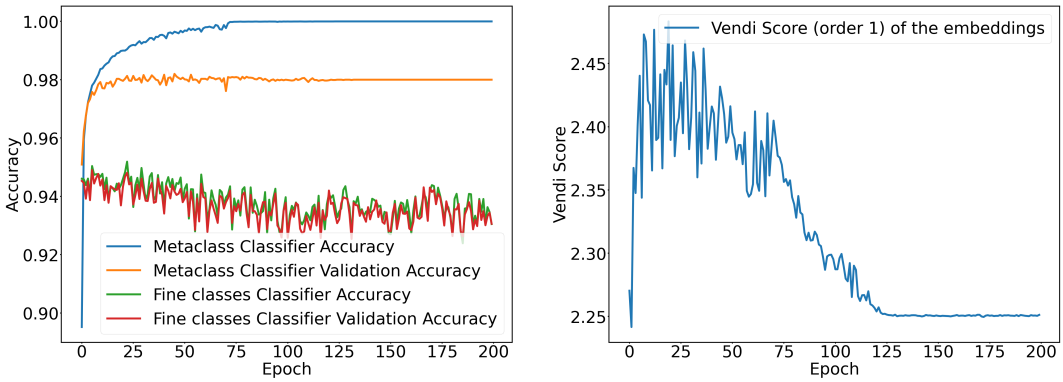

Figure 4: **Accuracy Scores and Vendi Score over training epochs on MNIST with batch size 256 and training loss as the monitor for the learning rate scheduler**.

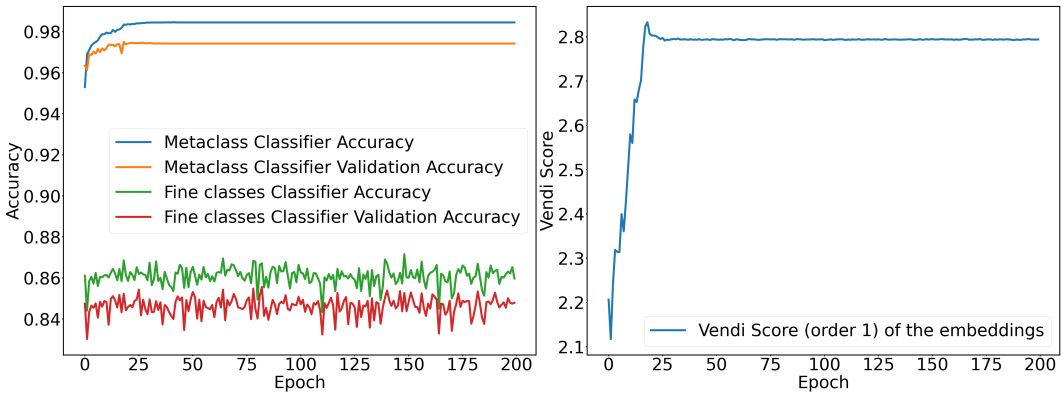

Figure 5: **Accuracy Scores and Vendi Score over training epochs on FASHION MNIST with batch size 256 and validation loss as the monitor for the learning rate scheduler**.

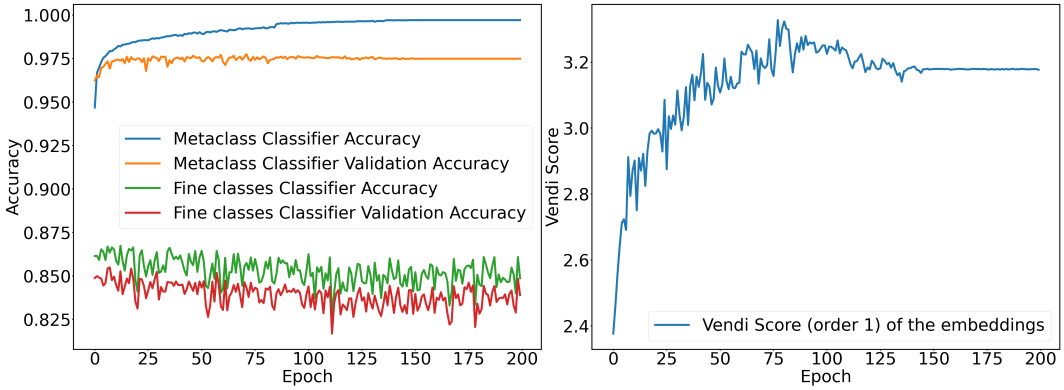

Figure 6: **Accuracy Scores and Vendi Score over training epochs on FASHION MNIST with batch size 256 and training loss as the monitor for the learning rate scheduler**.

From Figures 3, 4, 5 and 6, we observe that when training our network in the MNIST and FASHION MNIST datasets, independently of the specific parameters for the learning rate scheduler, the Vendi Score seems to be converging to a specific value throughout the training epochs, although with some continuous random spiking along the process, which is most likely due to the constant tweak in the network's parameters, as we can observe that this spiking is less noticeable as the learning rate scheduler keeps reducing the learning rate, until the reduction of the learning rate by the learning rate scheduler is enough to make the Vendi Score stabilize. When favoring the overfitting of the model, the Vendi Score stabilizes at a higher value than the number of metaclasses, although at a close enough value to consider the results fitting to our hypothesis. On the other hand, when favoring the generalization of the method, the value at which the Vendi Score stabilizes is

further from the number of metaclasses, meaning the embeddings being generated are more diverse. This comes with higher values for the accuracy of the model in the fine classes classification and the validation accuracy in the metaclasses classification, which as a whole indicates that a sufficiently low diversity is indeed a bottleneck for classification into any type of classes.
The plots corresponding to the experiments carried out with a batch size of 1024 can be found in the section Additional Plots of the Appendix as Figures 9-12.

		FASHION MNIST		MNIST	
		Batch size 256	Batch size 1024	Batch size 256	Batch size 1024
LRs monitor: Training loss	Validation accuracy	0,975	0,975	0,98	0,975
	Fine classification accuracy	0,86	0,865	0,94	0,95
	Vendi Score	3,2	2,6	2,25	2,5
	Epochs until convergence	150	115	125	140
LRs monitor: Validation loss	Validation accuracy	0,975	0,97	0,98	0,975
	Fine classification accuracy	0,86	0,86	0,95	0,95
	Vendi Score	2,8	2,18	3,35	2,5
	Epochs until convergence	25	50	25	40

Table 1: **Summary of the approximated values on stabilization from the experiments.** The table summarizes the approximate values for some of the metrics in our results once the learning rate scheduler has stabilized the training process on these. *LRs monitor* refers to the metric monitored by the learning rate scheduler. *Validation accuracy* and *Fine classification accuracy* refer to the accuracy of the network when classifying the test set into the metaclasses and the accuracy of the network when classifying the training set into the fine classes of the dataset respectively. *Epochs until convergence* refers to the number of epochs needed for the Vendi Score to stabilize as a consequence of the reduction of the learning rate.

The metric *Epochs until convergence* is included in Table 1 to emphasize that when setting the learning rate scheduler to focus on optimizing the training loss, it will take longer to stabilize, since we are essentially overfitting the training data, and we can thus keep improving the training loss every epoch, making the learning rate scheduler take longer to consider the improvement in the training loss insufficient between epochs and hence reducing the learning rate. When focusing on the validation loss, on the other hand, achieving a significant improvement between epochs is not as easy or common, making the learning rate scheduler reduce the learning rate in less epochs.
A fact worth taking into account is that when increasing the batch size for the training of the neural network, the diversity of the embeddings produced by the

network decreases, as we can observe in Table 1, where the Vendi Score is lower for the experiments with a batch size of 1024 than for the experiments with a batch size of 256.
The reason for this behavior when modifying the batch size is that the lower the batch size, the more precise are the changes made to the parameters of the network in each step of the backpropagation process, hence leading to a higher specificity when mapping the samples to the embeddings (e.g. less "collapse" of the embeddings) and a higher diversity among the embeddings.

Extra experiments on CIFAR-100

Once the experiments on the MNIST and FASHION MNIST datasets were carried out, we decided to replicate the procedure on the CIFAR-100 dataset, a more complex dataset as explained in section Description of the Used Datasets in the Appendix, in order to see how would the results be impacted by the use of a more complex dataset. In order to achieve significant results, we had to use a more complex neural network architecture, the efficientNetB0 architecture, which is a more sophisticated model originally proposed in [23]. However, as we will explain in the following paragraph, the results obtained from the experiments on the CIFAR-100 dataset were not quite satisfying.

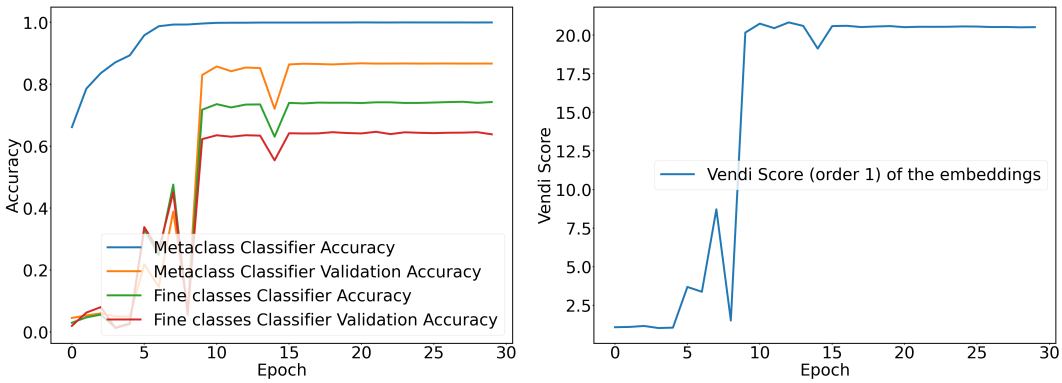

Figure 7: **Accuracy Scores (upper) and Vendi Score (lower) over training epochs on CIFAR-100 with batch size 32 and validation loss as the monitor for the learning rate scheduler**.

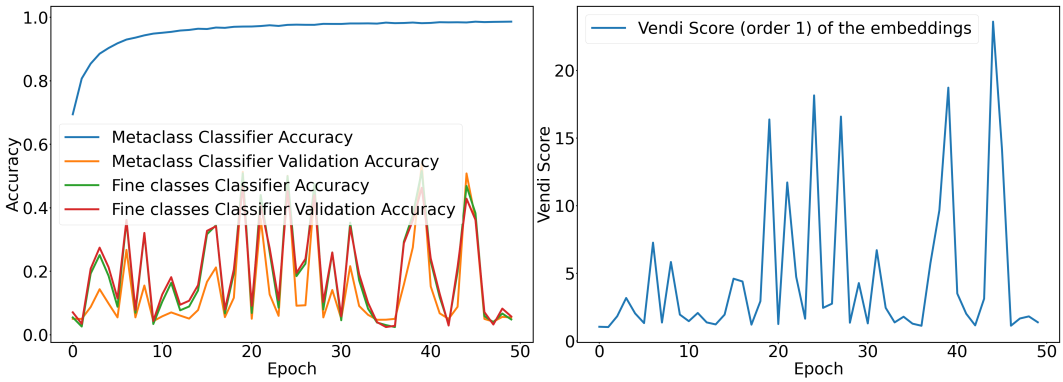

Figure 8: **Accuracy Scores and Vendi Score over training epochs on CIFAR-100 with batch size 32 and training loss as the monitor for the learning rate scheduler**.

As we can observe in Figure 7, the model stabilizes around epoch 15, which is a very low number of epochs compared to the experiments on the MNIST and FASHION MNIST datasets, and does not leave room for us to analyze the behavior of the Vendi Score before it is stabilized. On the contrary, in Figure 8, the model does not stabilize at all and all we are left is with a plot of the Vendi Score that is constantly spiking, which makes it impossible to extract any conclusions from it. What stands out the most from the results on CIFAR-100 (Figures 7 and 8) compared to the results in MNIST and FASHION MNIST (Figures 3-6) is that in the former, all the accuracy metrics except for the training accuracy have an identical behavior to the Vendi Score, being their plots almost a copy of the Vendi Score plot. The only explicable reason we could find behind this behavior is that since the classes and the neural network used with the CIFAR-100 dataset are more complex than the ones used with the MNIST and FASHION MNIST datasets, when training with the latter datasets and swapping the metaclass classifier output layer by the fine class classifier output layer, after training the fine class classifier output layer, this last output layer becomes more or less capable of distinguishing between the embeddings corresponding to samples from different fine classes without the need of the embeddings being very representative of these samples, while the same is not true for the CIFAR-100 dataset, where the embeddings need to be more representative of the samples in order to allow the fine class classifier output layer to be able to distinguish between the embeddings of the samples of the different fine classes. In other words, the diversity of the embeddings in CIFAR-100 is a more impactful and direct bottleneck for the accuracy of the network when classifying into the fine classes and when classifying the test set into metaclasses, acting as a limit for these metrics.

Chapter 5

Conclusions

As the reader may have noticed by now, although the project is about diversity metrics in deep learning embeddings, the main focus of the project is a single family of metrics, the Vendi Scores, while the rest of existing metrics for diversity in machine learning are just mentioned with the objective of providing the actual context for diversity in this field, which as we have seen, not much effort and research has been put into [20][21].
The Vendi Scores are a powerful tool for evaluating diversity not only in embeddings as it is the case in this project, but also in any other applications where measuring diversity is of interest, for example, in recommender systems [11].
Furthermore, the way the Vendi Score arises, taking ideas from ecology and quantum mechanics, always using mathematics as the common language, is a clear example of how interdisciplinary research can lead to the development of innovative and fruit-bearing ideas.

Despite not being able to find a direct relationship between the diversity of the embeddings generated by a neural network and the performance of the network when carrying out any task from the experiments carried out in this project, we can conclude that the diversity of the embeddings in a classifier neural network is related to the number of classes in which the network classifies the data. This is concluded from the fact that the Vendi Score of the embeddings stabilizes around the number of metaclasses of the dataset in the experiments carried out in this project, which is a clear indicator that the embeddings are more diverse when the network classifies the data into more classes.
Coming back to our initial objective for the project and from what we have discussed throughout Chapter 4 we can conclude that the possibility of using a neural network for a different task than the one it was trained for is not possible in practice. This is concluded from the fact that despite having two of the simplest

datasets available (MNIST and FASHION MNIST) and proposing basic classification tasks (classifying into 2 metaclasses and 10 fine classes) in our setting for the experiments, we were not able to achieve a satisfactory performance of the network when classifying into the fine classes of the dataset after training it for classifying into the metaclasses of the dataset.
Meaning that in a real case scenario, where the datasets are more complex and the tasks are more demanding, the performance of a model when carrying out a task it was not trained for would likely be even worse.

Besides the already mentioned current drawbacks of the Vendi Scores, future work on this project could involve using a sophisticated technique for enforcing diversity in the embeddings in order to favor the preservation of the diversity of the training samples and hopefully allowing the model to keep classifying the samples into the metaclasses with high accuracy without collapsing them into a type of embedding corresponding to the metaclass. If this were to be achieved, we hypothesize that the model would be able to generate embeddings as diverse as the samples in the training set, and would thus facilitate the task of classifying the samples into the fine classes with relatively high accuracy and no need for specific and thorough training.
As a matter of fact,[22] proposed *Vendi Sampling*, an algorithm for increasing the efficiency and efficacy of the exploration of molecules. In Vendi sampling, replicas are simulated in parallel and coupled via the Vendi Score, to enhance diversity. Therefore a similar approach could be used in our context to enforce diversity in the embeddings.

As an extra, it would be worth mentioning the main difficulties encountered during the development of the project. One was finding sources about deep learning embeddings which did not refer to word embeddings in natural language processing, which is the most common topic nowadays and as a consequence all search engines seem to prioritize this topic over everything else.
Besides this anecdotal problem, the main burden for the development of the project was the extraction of the results from the experiments, since the training of the neural networks was computationally expensive and time-consuming, and the results were not always as expected, which led to the need of repeating the experiments with different parameters and settings.

Bibliography

[1] Michael A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[2] Tom M Mitchell, *Machine Learning*, Vol. 1. McGraw-Hill New York, 1997.

[3] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza, *A survey and taxonomy of loss functions in machine learning*, arXiv preprint arXiv:2301.05579v1, (2023).

[4] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of Machine Learning Research, 12(Jul), 2121-2159, (2011).

[5] T. Tieleman and G. Hinton, *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural networks for machine learning, 4(2), 26-31, (2012).

[6] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[7] J. Schmidhuber, *Annotated History of Modern AI and Deep Learning*, Technical Report IDSIA-22-22, Swiss AI Lab IDSIA, USI & KAUST AII, (2022).

[8] J. Schmidhuber, *Deep Learning in Neural Networks: An Overview*, Technical Report IDSIA-03-14 / arXiv:1404.7828 v4 [cs.NE], The Swiss AI Lab IDSIA, University of Lugano & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland, Oct 2014.

[9] D. Bashir, G. D. Montañez, S. Sehra, P. Sandoval Segura, J. Lauw, *An Information-Theoretic Perspective on Overfitting and Underfitting*, arXiv preprint arXiv:2010.06076v2, (2020).

[10] E. Zvornicanin, G. Piwowarek, *What Are Embedding Layers in Neural Networks?*, Accessed on Web, (2024).

[11] Bondarenko, Kirill (2019), *Similarity metrics in recommender systems*, Accessed on Web, (2024).

[12] T. Leinster and C. A. Cobbold, *Measuring diversity: the importance of species similarity*, Ecology, 93(3), 477-489, (2012).

[13] P. R. Halmos, *Finite dimensional vector spaces*, Springer, ISBN 0-387-90093-4, (1974), Chapter 1, Section 25.

[14] T. Leinster, *Entropy and diversity: The axiomatic approach*, arXiv preprint arXiv:2012.02113, (2020).

[15] E. C. T. Inequalities. By G.H. Hardy, J.E. Littlewood and G. Pólya. 2nd edition. Pp. xii, 324. 27s. 6d. 1952. (Cambridge University Press). The Mathematical Gazette. 1953;37(321):236-236. doi:10.1017/S0025557200027455

[16] Anne Chao, Chun-Huo Chiu, and Lou Jost, *Phylogenetic Diversity Measures and Their Decomposition: A Framework Based on Hill Numbers*, Biodiversity Conservation and Phylogenetic Systematics, Topics in Biodiversity and Conservation, vol. 14, Springer International Publishing, pp. 141–172, (2016).

[17] M. Hill, *Diversity and evenness: a unifying notation and its consequences*, Ecology, 54(2), 427-432, (1973).

[18] R. H. Whittaker, *Evolution and measurement of species diversity*, Taxon, 21(2-3), 213-251, (1972).

[19] P. K. Jain and Khalil Ahmad, *5.1 Definitions and basic properties of inner product spaces and Hilbert spaces*, Functional Analysis (2nd ed.), New Age International, p. 203, ISBN 81-224-0801-X, (1995).

[20] Dan Friedman and Adji Bousso Dieng, *The Vendi Score: A Diversity Evaluation Metric for Machine Learning*, Department of Computer Science, Princeton University and Vertaix, arXiv:2210.02410v1 [cs.LG], (2022).

[21] Amey P. Pasarkar and Adji Bousso Dieng, *Cousins Of The Vendi Score: A Family Of Similarity-Based Diversity Metrics For Science And Machine Learning*, Department of Computer Science, Princeton University and Vertaix, arXiv:2310.12952v3, Published in Artificial Intelligence and Statistics, AISTATS 2024, May 7, 2024.

[22] Amey P. Pasarkar, Gianluca M. Bencomo, Simon Olsson, and Adji Bousso Dieng, *Vendi Sampling For Molecular Simulations: Diversity As A Force For Faster Convergence And Better Exploration*, Department of Computer Science,

Princeton University, Department of Computer Science and Engineering, Chalmers University, and Vertaix, arXiv:2307.00345v1 [cond-mat.stat-mech], (2023).

[23] M. Tan and Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, arXiv preprint arXiv:1905.11946v5 [cs.LG], Sep 2020.

[24] Shen, T., Ott, M., Auli, M., and Ranzato, M. (2019). Mixture models for diverse machine translation: Tricks of the trade.*In International conference on machine learning*, pages 5719–5728. PMLR.

[25] Fomicheva, M., Sun, S., Yankovskaya, L., Blain, F., Guzmán, F., Fishel, M., Aletras, N., Chaudhary, V., and Specia, L. (2020). *Unsupervised Quality Estimation for Neural Machine Translation*. Transactions of the Association for Computational Linguistics, 8:539–555.

[26] Benhenda, M. (2017). *ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity?* arXiv preprint arXiv:1708.08227.

[27] Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. (2018). *Assessing Generative Models via Precision and Recall*. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 5234–5243.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, in Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, arXiv:1706.08500v6 [cs.LG], Jan 2018.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved Techniques for Training GANs*, arXiv preprint arXiv:1606.03498, (2016).

[30] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). *VEEGAN: reducing mode collapse in GANs using implicit variational learning*. In Advances in Neural Information Processing Systems.

[31] Bullen, P. S. (2003). *Handbook of Means and Their Inequalities*. Dordrecht, Netherlands: Kluwer, pp. 175-177.

Appendix

Description of the Used Datasets

Description of MNIST dataset

The MNIST dataset is a dataset of 28x28 gray-scale images of handwritten digits (0-9), which make up the 10 fine classes. For the metaclasses, we will consider the even and odd digits as two metaclasses.

It contains 60,000 training images and 10,000 testing images, with each class having 6,000 training images and 1,000 testing images.

Description of Fashion MNIST dataset

The Fashion MNIST dataset is a dataset of 28x28 gray-scale images of clothing items, including 10 different classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. For the metaclasses, we will consider the items of clothing that are worn on the upper body as one metaclass (this includes T-shirt/top, Pullover, Coat, Shirt, Bag) and the items of clothing that are worn on the lower body as another metaclass (this includes Trouser, Dress, Sandal, Sneaker, Ankle boot).

It contains 60,000 training images and 10,000 testing images, with each class having 6,000 training images and 1,000 testing images.

Description of CIFAR-100 Dataset

The CIFAR-100 dataset is a dataset of 32x32 color images of 100 different fine classes and 20 coarse classes or metaclasses.

It contains 50,000 training images and 10,000 testing images, with each fine class having 500 training images and 100 testing images and thus each metaclass having 2,500 training images and 500 testing images.

Some examples of the metaclasses are: large carnivores, large omnivores and herbivores, medium-sized mammals, trees, vehicles 1, vehicles 2, etc.

Some examples of the fine classes are: beaver, dolphin, otter, seal, whale, aquarium fish, flatfish, ray, shark, trout, orchids, poppies, roses, sunflowers, tulips, bottles, bowls, cans, cups, plates, apples, mushrooms, oranges, pears, sweet peppers, clocks, computers, keyboards, telephones, televisions, etc.

Additional plots

Here we present the plots obtained from the experiments that were not included in the main body of the document for them being redundant or not adding any new information to the discussion. Nevertheless, these plots are all summarized among others in Table 1 of Chapter 4.

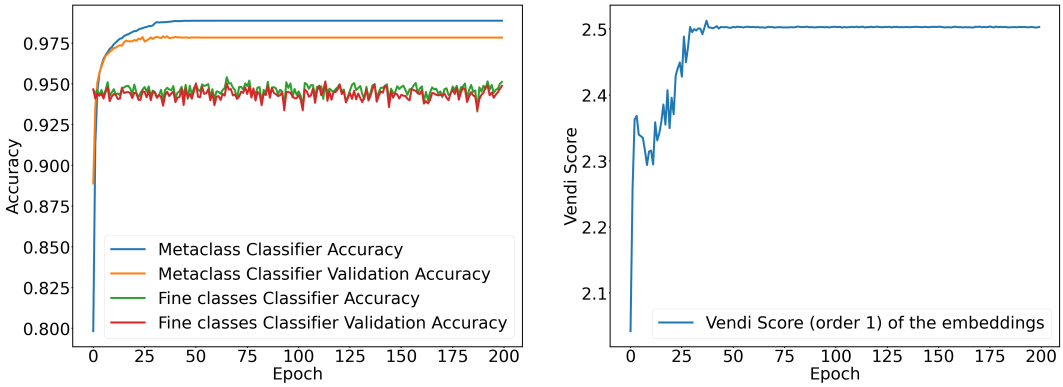

Figure 9: **Accuracy Scores and Vendi Score over training epochs on MNIST with batch size 1024 and validation loss as the monitor for the learning rate scheduler.**

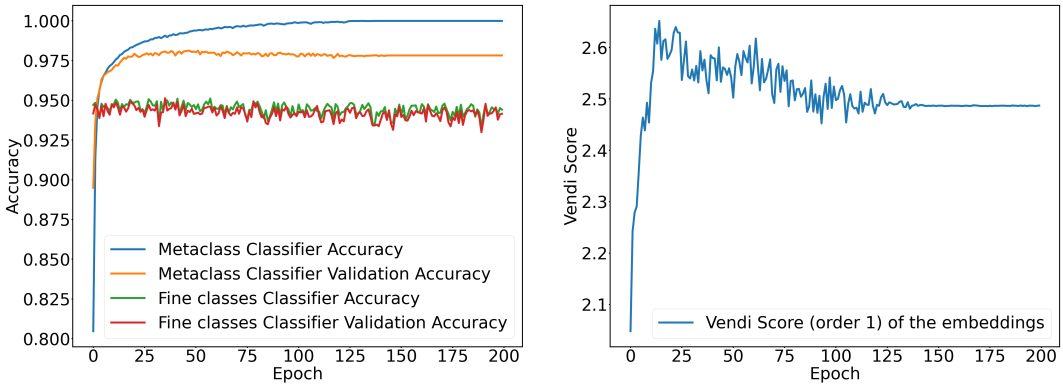

Figure 10: **Accuracy Scores and Vendi Score over training epochs on MNIST with batch size 1024 and training loss as the monitor for the learning rate scheduler.**

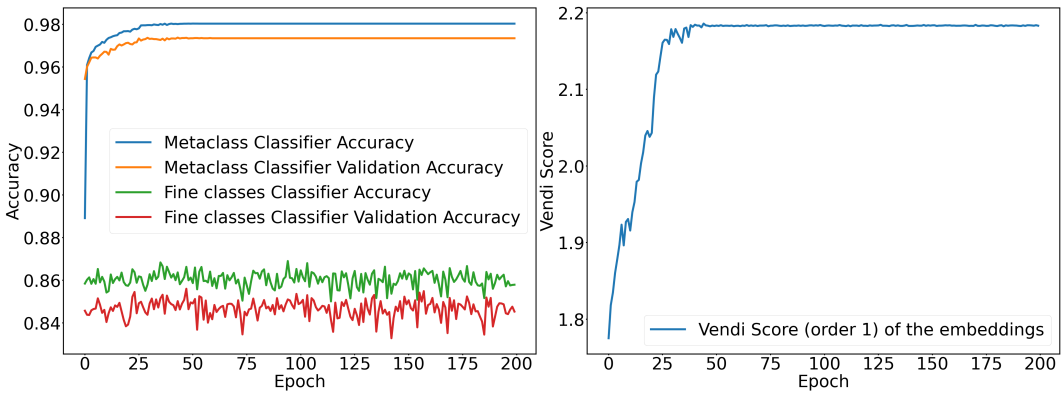

Figure 11: **Accuracy Scores and Vendi Score over training epochs on FASHION MNIST with batch size 1024 and validation loss as the monitor for the learning rate scheduler.**

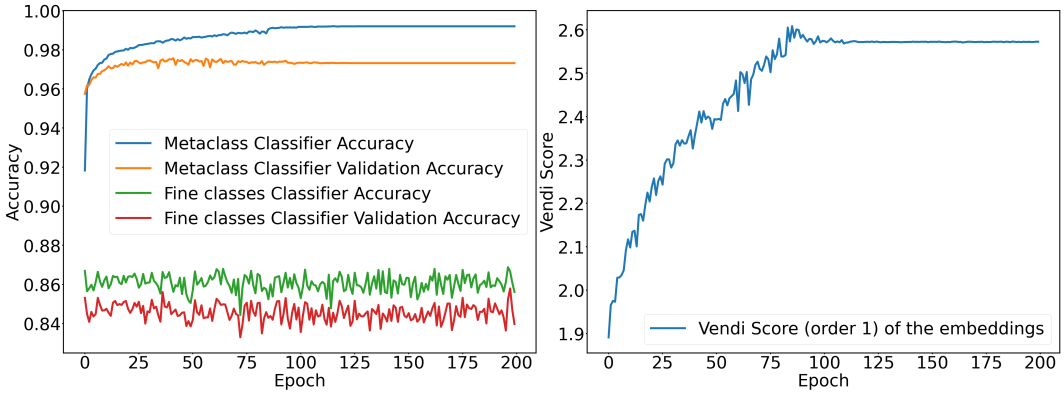

Figure 12: **Accuracy Scores and Vendi Score over training epochs on FASHION MNIST with batch size 1024 and training loss as the monitor for the learning rate scheduler.**