Predicting the generalization gap in neural networks using topological data analysis

Rubén Ballester[a,b,1,*], Xavier Arnal Clemente[a], Carles Casacuberta[a,2], Meysam Madadi[b], Ciprian A. Corneanu[c], Sergio Escalera[a,b,3]

[a] *Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Gran Via de les Corts Catalanes 585, 08007 Barcelona, Spain*
[b] *Computer Vision Center, Campus UAB, 08193 Bellaterra, Barcelona, Spain*
[c] *Amazon Apollo, 25 9th Ave N, Seattle, WA 98109, United States*

Abstract

Understanding how neural networks generalize on unseen data is crucial for designing more robust and reliable models. In this paper, we study the generalization gap of neural networks using methods from topological data analysis. For this purpose, we compute homological persistence diagrams of weighted graphs constructed from neuron activation correlations after a training phase, aiming to capture patterns that are linked to the generalization capacity of the network. We compare the usefulness of different numerical summaries from persistence diagrams and show that a combination of some of them can accurately predict and partially explain the generalization gap without the need of a test set. Evaluation on two computer vision recognition tasks (CIFAR10 and SVHN) shows competitive generalization gap prediction when compared against state-of-the-art methods.

Keywords: Deep learning, neural network, topological data analysis, generalization gap

1. Introduction

Understanding the generalization capacity of a neural network is one of the most important questions in deep learning. Unfortunately, while the fundamental procedures of training neural networks are well understood, being able to tell why one network is better at generalizing than another still poses a great challenge. Good performance of a deep neural network (DNN) depends fundamentally on its architecture and its neuron functions and parameters. These yield

*Corresponding author

Email addresses: ruben.ballester@ub.edu (Rubén Ballester), xavi.aclm@gmail.com (Xavier Arnal Clemente), carles.casacuberta@ub.edu (Carles Casacuberta), meysam.madadi@gmail.com (Meysam Madadi), cipriancorneanu@gmail.com (Ciprian A. Corneanu), sergio.escalera.guerrero@gmail.com (Sergio Escalera)

[1] Supported by the Ministry of Universities of Spain through contract FPU21/00968

[2] Partially supported by the Ministry of Science and Innovation of Spain through project PID2020-117971GB-C22 and by the Generalitat de Catalunya with reference 2021 SGR 00697

[3] Partially supported by the Ministry of Science and Innovation of Spain through PID2022-136436NB-I00 and by the ICREA Acadèmia programme

an approximation of the desired function (prediction or regression) based on neuron interactions —the better the approximation, the better the generalization. However, with the high quantity of neurons and connections of deep neural networks (sometimes of the order of millions), understanding which interactions between neurons are improving or damaging a model is a hard problem. Developing new mathematical tools that capture the effect of these interactions on the output of the networks is key for increased understanding of network generalization.

A DNN that generalizes will perform well on test data on which it has not been trained. This is usually measured by the generalization gap, which is defined as the difference between the accuracy in training versus test datasets. Although the two accuracies are correlated to a certain extent, studying training performance alone can be misleading. Several papers show how neural network performances on unseen examples can differ with respect to their training performances due to many reasons [1, 2, 3]. *To what extent is it possible to predict the generalization gap without testing a model?* In a practical sense, a measure of generalization that does not require a test dataset eliminates the responsibility of maintaining and curating such a dataset.

The issue of finding a generalization measure has been explored extensively and a recent challenge on the topic provides an excellent framework for algorithmic benchmarking [4]. However, the most competitive participant methods rely on internal representations of independent layers, discarding more global structures that may be created across the network [5, 6] or even discarding structure altogether [7].

An alternative approach is provided by topological data analysis (TDA), an applied branch of algebraic topology that studies the shape of sets of points endowed with a metric structure. Such shapes are described by means of persistence diagrams [8], which are built on homological features of simplicial complexes constructed from the given dataset.

In this paper we present an approach to predict the generalization gap from persistence diagrams based on neuron interactions in deep neural networks of any size. For this purpose, we use weighted graphs computed from activation correlations between neurons after training a network with a dataset. We compare the performance of different persistence diagram vectorizations, called persistence summaries, from which the generalization gap can be regressed, and we find that a suitable combination of such summaries yields competitive results on measuring the generalization gap. Moreover, we show that persistence summaries separate neural network architectures into clusters related with their generalization capacity.

The findings from this study were applied in a series of proof-of-concept experiments detailed in [9], which involved the creation of regularizers for deep learning models. The TDA-based regularizers introduced in [9] aimed to diminish the most significant correlations among neurons while still maintaining a degree of redundancy. By doing so, they surpassed the performance of several commonly utilized regularizers in minimizing generalization gaps.

The paper is structured as follows: in Section 2 we discuss related work; in Section 3 we define functional graphs and describe their persistence summaries; in Section 4 we present and discuss experimental results, and conclusions are written down in Section 5. Supplemental material is provided in an appendix.

The code for this article is available in the following repository:
https://github.com/rballeba/PredictingGeneralizationGapUsingPersistentHomology

2. Related Work

Predicting generalization. Understanding the generalization gap is a major area of research in theoretical and practical deep learning. One of the most influential papers in the last few years has been [10], in which classical theories on the generalization capabilities of machine learning models were shown to fail to explain why neural networks generalize well in practice. This paper motivated a tremendous amount of original work on generalization of deep neural networks. From the theoretical point of view, some works tried to correct the flaws of the previous methods by developing new and tighter generalization bounds [11, 12, 13, 14, 15, 16, 17, 18, 19, 20], by studying generalization measures [21, 22], or by studying the training process [23, 24, 25], among others. From an experimental point of view, there have been many works studying generalization measures and trying to predict the generalization gap. One of the most extensive benchmarks for the robustness of generalization measures was developed in [26], where 40 different generalization measures were tested in more than $10{,}000$ trained models. With the objective of developing new robust generalization measures, the first competition on predicting generalization in deep learning (PGDL) was organized at NeurIPS [4] and its results were published in [27]. The generalization measures presented there were divided into three main categories: 1. Measures based on theoretical generalization bounds; 2. Measures based on data augmentation; and 3. Measures based on the analysis of intermediate representations. The winners of the competition, the teams Interpex [6], Always Generalize [7], and BrAIn [5], presented generalization measures in the last two categories. The Interpex team proposed a generalization measure based on neuroscience ideas that uses the Davies–Bouldin index [28] to quantify the consistency of internal representations of neural networks, the Always Generalize team proposed to measure the robustness of neural networks against data-augmented datasets, and the BrAIn team proposed a measure based on properties of a graph constructed from the internal representations of a neural network. After the competition, other robust generalization measures were published [29, 30].

The lack of winners based on theoretical generalization bounds suggests that theoretical bounds are still far from being usable in practical scenarios, and that new and original methods are needed to keep improving our understanding of the generalization phenomenon in neural networks. On this aspect, our approach, while being novel in its methodology, obtains state-of-the-art performance when predicting the generalization gap compared with the winning methods of the PGDL competition.

Topological data analysis. Topological data analysis (TDA) has been used very successfully in machine learning. A survey of applications is offered in [31]. From a theoretical point of view, topological data analysis has been used to analyze structural properties of neural networks [32], input and output spaces [33, 34, 35, 36, 37, 38, 39], generative models and their properties [40, 41], and internal representations and weights of neural networks [42, 43, 44, 45, 46], among others.

In the intersection of topological data analysis with prediction of the generalization gap, we find [47, 48, 49]. In [47], a novel connection between the upper box dimension and the persistent homology dimension [50, 51] is used to bound the generalization gap of neural networks using the fractal dimension of training trajectories [19]. In [48, 49], generalization of neural networks is studied by calculating persistent homology of activation vectors of the neural network on the training dataset. In particular, in [49], the generalization gap is predicted with linear models based on persistence summaries extracted from neuron activations.

However, the existing methods fail to be suitable in certain scenarios. On the one hand, [47] cannot be used without the training information of a neural network, which is generally not available when using pretrained models. On the other hand, the methods from [48, 49] do not scale to modern neural network architectures, as they compute descriptors from persistence diagrams, which share in most cases a computational complexity higher than cubical on the number of neurons in the network. In addition, the summaries of persistence diagrams tested in these articles are scarce, and other persistence summaries could potentially be better suited to predict the generalization gap.

2.1. Contributions

In this article, the following contributions are made:

1. We extend the methodology of [48, 49] to make it capable of processing large neural networks. To achieve this, we propose a methodology that performs bootstrapping on persistence summaries computed from persistence diagrams coming from different samples of neurons from the same network. Samples are taken following a probability distribution over the set of neurons of the network, giving more probability to neurons that share high activation values.

2. We train linear models using eleven different combinations of persistence summaries to predict the generalization gap and compare these models with linear models trained on the generalization measures proposed by the PGDL competition winners [4], obtaining competitive results. We find that basic statistical parameters of the distribution of points in persistence diagrams are the best performing summaries to predict generalization gaps.

3. We offer an interpretation of why topological data analysis is meaningful for predicting the generalization gap from features learned by a neural network. Figure 4.3 illustrates a neat clustering phenomenon of network architectures with respect to their depth when the generalization gap is represented in relation with suitable persistence summaries.

3. Methodology

Let $N\colon \mathcal{X} \to \mathcal{Y}$ denote a classification neural network, where $\mathcal{X}$ is a set of inputs and $\mathcal{Y}$ is a set of labels. Let $\mathcal{L}$ be a loss function on $\mathcal{Y} \times \mathcal{Y}$ that measures the error of a prediction, and let $\mathcal{D}, \mathcal{T} \subseteq \mathcal{X} \times \mathcal{Y}$ be a pair of training and test datasets, respectively. Let

$$\mathcal{R}[N] = \mathbb{E}_{(x,y)\sim\mathbb{P}_{(X,Y)}}[\mathcal{L}(N(x), y)]$$

be the expected risk of N, where $\mathbb{P}_{(X,Y)}$ is a generally unknown data distribution, and let

$$\mathcal{R}_S[N] = \frac{1}{|S|}\sum_{i=1}^{|S|}\mathcal{L}(N(x_i), y_i)$$

be the empirical risk function on a dataset $S = \{(x_i, y_i)\} \subseteq \mathcal{X} \times \mathcal{Y}$.

A main objective in classification tasks is to find an optimal network N_{opt} from a given set of neural networks that minimizes the expected risk $\mathcal{R}[N]$. In most cases, $\mathcal{R}[N]$ cannot be computed, since the data distribution function $\mathbb{P}_{(X,Y)}$ is not known. Therefore, the usual approach is to minimize the empirical risk function $\mathcal{R}_{\mathcal{D}}[N]$ using the training dataset $\mathcal{D}$.

In the special case of the 0-1 loss function $\mathcal{L}(\hat{y}, y) = 1$ if $\hat{y} = y$ and 0 otherwise, the empirical risk can be written as $\mathcal{R}_{\mathcal{D}}[N] = 1 - \text{Acc}_{\mathcal{D}}[N]$, where $\text{Acc}_{\mathcal{D}}[N]$ is the training accuracy used as benchmarking measure in most deep learning classification problems. Therefore, minimizing the empirical risk for this function $\mathcal{L}$ is equivalent to maximizing the training accuracy.

However, minimization of empirical risks does not necessarily lead to minimization of expected risks, due to phenomena such as overfitting. The difference $\mathcal{R}[N] - \mathcal{R}_{\mathcal{D}}[N]$ between both quantities is known as the *generalization gap* of the neural network N. This quantity is usually approximated with the *empirical generalization gap*, which is defined as the difference $\mathcal{R}_{\mathcal{T}}[N] - \mathcal{R}_{\mathcal{D}}[N]$ between the empirical risks for the training and test datasets. For the 0-1 loss function, the empirical generalization gap is equal to the difference $\text{Acc}_{\mathcal{D}}[N] - \text{Acc}_{\mathcal{T}}[N]$ between the accuracies in train and in test. With the realistic assumption that current neural networks obtain better training accuracy than test accuracy and that training accuracies are generally high, a lower generalization gap is an indication of a better network performance.

3.1. Objectives

The main purpose of this paper is to predict the empirical generalization gap using only information from the training dataset $\mathcal{D}$ by gleaming information about the dynamic behaviour of a trained neural network, i.e., the internal representations, structures and relationships between neuron activations during classification. In our context, the network behaves dynamically only in the presence of input data, forming a graph of neuron activations.

Our first goal is to define a mathematical structure describing the activation of a network when fed with a specific dataset $\mathcal{D}$ consisting of pairs (x, y) where x and y represent inputs and ground truth annotations respectively. To do so, we use a complete weighted graph whose set of vertices is in bijective correspondence with the set of neurons of the given network. Each vertex in this graph is represented by an activation vector of dimension $|\mathcal{D}|$ where the vector components are the neuron's activations for all $(x, y) \in \mathcal{D}$. Edges are weighted by a correlation distance between the activation vectors that they are connecting.

From this weighted graph we build a filtered simplicial complex computed from the edge weights, whose topological features are described by a persistence diagram, from which we extract suitable summaries with the purpose of relating them with the empirical generalization gap of the network. Precise definitions are given in the next subsections.

3.2. Network functional graphs

Let $V = \{v_1, \ldots, v_n\}$ be the set of non-input nodes of a neural network N trained with a dataset $\mathcal{D} = \{(x, y)\}$, where x denotes inputs and y denotes corresponding values from a set of labels. For a node $v \in V$, we denote by $N_v(x)$ the activation value of v on some input x, and define the *activation vector* of v as

$$A_v(\mathcal{D}) = (N_v(x))_{(x,y)\in\mathcal{D}}.$$

The set $A_N(\mathcal{D}) = \{A_v(\mathcal{D}) \mid v \in V\}$ of activation vectors is meant to capture the role of each node of N during inference.

A *correlation distance* between two nodes $v_i, v_j \in V$ is defined as

$$d(v_i, v_j) = 1 - |\text{corr}(A_{v_i}(\mathcal{D}), A_{v_j}(\mathcal{D}))|, \tag{3.1}$$

where corr is the Pearson correlation coefficient. Nodes with constant activations can be safely regarded as not affecting the behaviour of the model, but rather its structure as a bias. Therefore,

nodes with zero variance are discarded. Although this function d does not satisfy the axioms of a metric, it is suitable for the application of techniques from TDA —this fact is discussed in Section 3.3.2 below.

The complete weighted graph with vertices the nodes in V with nonzero variance and weights $d(v_i, v_j)$ on the edges will be called the *functional graph* of the trained neural network N. This graph encodes the functional behaviour of N. In this article we use Vietoris–Rips filtrations associated with the distance matrix $(d(v_i, v_j))$ from the functional graph for a homological persistence study, as defined in the next section.

3.3. Topological Data Analysis

3.3.1. Vietoris–Rips complexes

An abstract simplicial complex, a basic tool of algebraic topology, is a finite collection of sets S such that if $\alpha \in S$ and $\beta \subseteq \alpha$ then $\beta \in S$. Each abstract simplicial complex K determines a sequence of *homology groups* $H_n(K)$ for $n \geq 0$, generated by linearly independent n-dimensional cycles modulo boundaries. In this article coefficients of homology groups are meant in the field $\mathbb{F}_2$ of two elements.

If V is a finite set equipped with a distance function d, then for each subset $\alpha \subseteq V$ we may consider the diameter $\mathrm{diam}(\alpha) = \max_{i,j\in\alpha} d(i, j)$ of α relative to d. The *Vietoris–Rips complex* of V at a parameter value $r \geq 0$ is an abstract simplicial complex defined as

$$\mathrm{VR}_r(V) = \{\alpha \subseteq V : \mathrm{diam}(\alpha) \leq r\}.$$

The set $\{\mathrm{VR}_r(V)\}_{r\geq 0}$ is a nested collection of simplicial complexes, as $\mathrm{VR}_r(V) \subseteq \mathrm{VR}_s(V)$ if $r \leq s$. Each such filtration yields a *persistence diagram* for every integer $n \geq 0$, which contains a point (r, s) for each homology generator of dimension n born at a parameter value r and vanishing at s, where $r < s$. Further details about persistence diagrams can be found in [8].

3.3.2. Correlation distance

The correlation distance d defined in (3.1) can take a zero value on distinct nodes and the triangle inequality need not hold. However, Vietoris–Rips filtrations can be associated with arbitrary functions $X \times X \to \mathbb{R}$ where X is any set, and stability holds in such generality [52, 53].

Although d does not necessarily satisfy that $d(x, y) \neq 0$ whenever $x \neq y$, this does not affect persistent homology, since the matrix $(d(v_i, v_j))$ yields Vietoris–Rips complexes homotopy equivalent to those obtained by identifiying two nodes x and y if $d(x, y) = 0$. Moreover, while d does not satisfy the triangle inequality, the following transformation does:

$$d'(v_i, v_j) = \sqrt{1 - (1 - d(v_i, v_j))^2}.$$

Since the function $\gamma(t) = \sqrt{1 - (1 - t)^2}$ is monotonic on $[0, 1]$ and uniformly continuous, d and d' produce the same Vietoris–Rips filtrations, albeit at different thresholds, and share similar continuity properties with respect to small displacements in the space of functional graphs.

3.3.3. Persistence summaries

There is a variety of numerical or vector-valued functions defined on persistence diagrams available for statistical analyses. We refer to such functions as *persistence summaries* or *descriptors*. In this subsection we present the summaries that have been used in our work.

Average and standard deviation of lifetime parameters. Different combinations of birth parameters and death parameters have been explored in this article, including their squares and the transformation $1/x + \ln x$ applied element-wise. We used averages and standard deviations of births and deaths as predictors of the generalization capacity of a network.

The *life* or *lifetime* of a point (b, d) in a persistence diagram is defined as $d - b$, while the *midlife* is $(b+d)/2$. Average lives and average midlives also yield useful results when predicting generalization gap using linear extrapolations; these summaries have been used previously with a similar purpose in [49]. Standard deviation or variance of lives and midlives work equally well or better. This technique is based on the heuristic that the generalization gap of a network is influenced by the average position and dispersion of points in persistence diagrams.

Persistence entropy. The definition of persistence entropy is an adaptation of the concept of entropy used in information theory, which, according to [54], provides a measure of the uncertainty of some random variable. The *entropy* of a persistence diagram P is defined as

$$\epsilon(P) = -\sum_{(b,d)\in P} ((d-b)/L)\ \log_2((d-b)/L)), \tag{3.2}$$

where $L = \sum_{(b,d)\in P} (d-b)$. If one defines a discrete random variable that picks points (b, d) from P weighted according to their life, then the persistence entropy corresponds to the entropy of this random variable. This choice of weights is based on the assumption that points near the diagonal carry less information. More details on persistence entropy can be found in [55].

Persistence pooling vectors. Persistence pooling vectors were introduced in [56] in order to improve a max-pooling procedure using TDA. This approach consists of analyzing only the most important points in a given persistence diagram, where importance is weighted according to the difference $d - b$. We define the n-th persistence pooling vector as the vector in descending order of the n maximum life values. If the persistence diagram has less than n points, then the void vector components are set to 0. We selected the highest 10 life values. This number has been chosen experimentally in view of the lack of score performance observed when selecting a larger number of vector components.

Complex polynomials. The persistence summary introduced in [57] transforms persistence diagrams into polynomials with coefficients in the field $\mathbb{C}$ of complex numbers whose roots are the images of persistence diagram points under a well-chosen mapping from $\mathbb{R}^2$ to $\mathbb{C}$. In our study we used the transformation T defined in [57].

4. Results

In the first part of this section, we describe experimental setups and comment on computational complexity (4.1). In the second part, we evaluate our approach and discuss results (4.2).

4.1. Experiments

Datasets. We use the dataset of trained DNNs provided by the *Predicting Generalization In Deep Learning* (PGDL) competition [4]. The dataset is divided into eight tasks, each composed of several neural network architectures trained to provide different generalization gaps on a particular dataset. We focus on the first two tasks, which were public when the competition was launched.

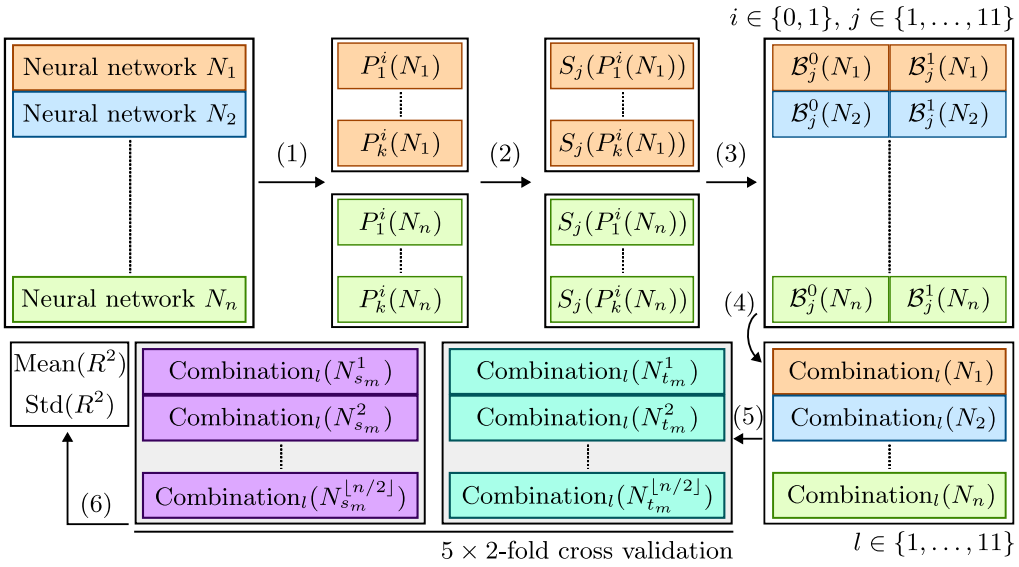

Figure 4.1: Experimental evaluation pipeline. Given a specific PGDL task as described in Section 4.1, let $\{N_i\}_{i=1}^n$ be the set of neural networks associated with the task. (1) Generation of k different persistence diagrams per DNN and dimension $i \in \{0,1\}$ using sampling in CIFAR10/SVHN datasets as described in Section 4.1.1. In our case, $k = 20$. (2) Computation of persistence summaries S_j introduced in Section 3.3.3 for each persistence diagram. (3) Bootstrapping for each dimension and each summary computed from the same DNN. The bootstrapped summary S_j for dimension i and neural network N is denoted by $\mathcal{B}_j^i(N)$. (4) Generation of the eleven different combinations of bootstrapped persistence summaries described in the experimental procedure of Section 4.1. (5) A 2-fold cross-validation partition into sets with the same cardinality is calculated five times. Each time, for each combination of summaries $l \in \{1,\dots,11\}$, two linear models to predict the generalization gap are trained on one of the partition sets and tested on the other, obtaining a R^2 score for each model on the test set. (6) We compute the mean and standard deviation of the resulting R^2 scores. Next, using the same partition sets, we train linear models with the generalization measures of the three winners of the PGDL competition, and we compare our best performing methods with their methods using 5×2-fold cross-validation statistical tests.

The first task consists of 96 VGG-like [58] neural networks, namely one for each combination of the following hyperparameters: each network is designed with either two or six convolutional layers and one or two dense layers; either 256 or 512 filters in the last convolutional layer; a dropout probability, chosen to be 0 or 0.5; a number of convolutional blocks in each convolutional layer, either one or three; the weight decay during training, set up to be 0 or 0.001; and the batch size, varying between 9, 32, and 512. Each network was trained on the CIFAR10 dataset [59], consisting of 60,000 color images of size 32×32 split into ten classes that represent vehicles (airplanes, automobiles, ships and trucks) and animals (birds, cats, deers, dogs, frogs, and horses).

The second task is composed of 54 neural networks with *network in network* architectures [60], with a varying number of convolutional layers. Specifically, each network is chosen with either six, nine, or twelve convolutional layers and trained on the SVHN dataset [61], a digit classification benchmark dataset that contains 600,000 color 32×32 images of printed digits (from 0 to 9) cropped from pictures of house number plates. Other hyperparameters of Task 2 networks are the dropout probability, chosen among 0, 0.25, and 0.5; the weight decay, either 0 or 0.001; and the batch size, varying between 32, 512, and 1024.

Experimental procedure. Our experimental procedure is depicted in Fig. 4.1, while the hyperparameters of our method are detailed as follows. Initially, we generate 20 distinct persistence diagrams for dimensions zero and one for each neural network, using the sampling methods outlined in Section 4.1.1. Subsequently, for each persistence diagram, we calculate persistence summaries as introduced in Section 3.3.3. This results in 20 distinct instances of each persistence summary for each network and homology dimension. Following this, we perform bootstrap analyses on each set of 20 values of persistence summaries derived from the same network and homology dimension. The bootstrapping process involves creating 1,000 bootstrap samples of size 20 selected with replacement from the various persistence summaries.

We combine bootstrapped persistence summaries to use them as predictor variables of the generalization gap with a linear regression for both tasks. The list of persistence summaries that we test is the following: (1) Persistence pooling of 10 elements; (2) average lives and average midlives; (3) average births and average deaths; (4) average and standard deviation of births and deaths; (5) persistence entropy; (6) complex polynomials with 10 coefficients.

We also test concatenations of (2), (3) and (4) with their element-wise squared versions, and a concatenation of (3) with its element-wise logarithmic version, as well as a concatenation of (2) and (3), original and squared. All combinations are considered in homological dimension zero, homological dimension one, and a concatenation of both.

We compare our models with linear regressions trained from the three generalization measures that won the PGDL competition; see Section 2 for further details on these generalization measures.

Evaluation metrics. We train linear regression models with the previous combinations of persistence summaries and the three state-of-the-art generalization measures to predict the generalization gap of neural networks. To measure and compare their performance, we use a 5×2-fold cross-validation statistical test, as recommended in [62], with the coefficient of determination R^2 as performance metric. The coefficient of determination R^2 is computed as the proportion of the variation in the dependent variable that can be predicted from the independent variables, and it is calculated as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4.1}$$

where y denotes the ordered set of actual values, $\hat{y}$ denotes the ordered set of predicted values, and $\bar{y}$ denotes the mean of y. This coefficient ranges from 0 to 1 in the training dataset but can be outside that range in unseen data. When the score is 1, the model perfectly predicts the values of y. A score of $R^2 = 0$ is obtained when one uses a horizontal line at the average of the set of y-values as a model. If a model performs worse than this (which usually indicates that the choice of model itself was ill-advised), then the numerator of (4.1) can grow arbitrarily large, and thus R^2 can be negative. If an R^2 value is negative, then the prediction is worse than ignoring the input and predicting the average of the sample. This can actually happen when the training set yields a model that does not generalize in the test set.

The 5×2-fold cross-validation statistical test validates if there are significant differences between two models tested in a common dataset. The null hypothesis of this test is that, for a fixed-size random drawn training dataset, two learning algorithms have the same R^2 score on a randomly drawn test dataset. We compare linear models pairwise for each task.

4.1.1. Reducing computational complexity

Computational complexity. Computing topological summaries with the complete set of activations calculated from the entire training dataset is unfeasible due to the high computational time and memory complexities of obtaining activation vectors and persistence diagrams. If $|\mathcal{D}|$ denotes the number of input samples for a dataset $\mathcal{D}$ and $|V|$ is the number of nodes in a neural network N, then the set of activation vectors of nodes in N for the dataset $\mathcal{D}$ has cardinality $|A_N(\mathcal{D})| = |\mathcal{D}| \times |V|$ (see Section 3.2 for details). Assuming that we have a standard current neural network like VGG16, that has about $8{,}000$ neurons [63] only for fully connected layers, a standard dataset like CIFAR10 [59] with $50{,}000$ training examples, and a double precision floating point format to represent each number, one would need at least 3 GB only to store the activations of fully connected layers. Additionally, although zero dimensional persistent homology can be calculated in $\mathcal{O}\left(|V|^2 \cdot A^{-1}(|V|^2)\right)$ using the algorithm proposed in [64] where A^{-1} is the notoriously slowly growing inverse of the Ackermann function [65, Chapter 21], persistent homology in higher dimensions is harder to compute. The complexity of algorithms for computing persistent homology for dimension greater than or equal to one is $O(n^3)$ if n is the number of simplices of the Vietoris–Rips complex and Gaussian elimination is used to find ranks of matrices of boundary operators, or $O(n^\omega)$ where ω is the exponent of matrix multiplication (currently 2.3729) if sparsity of boundary matrices is taken into account, as in [66]. In its turn, the number of simplices n depends cubically on the number $|V|$ of vertices of the functional graph if persistence diagrams are drawn only in homological dimension one, which requires determination of simplices up to dimension two.

In practice, this limits persistence diagram computations to a few thousand vertices. In order to alleviate these problems in neural networks with a large set of neurons, we introduce sampling strategies for both the input dataset and the functional graphs.

Sampling the input space. We compute activation vectors A_v for a fixed subsample $\mathcal{D}' \subseteq \mathcal{D}$. In order to justify that this subsampling does not affect the results of the analysis, it is enough to verify that $\operatorname{corr}(A_{v_i}(\mathcal{D}'), A_{v_j}(\mathcal{D}'))$ is sufficiently close to $\operatorname{corr}(A_{v_i}(\mathcal{D}), A_{v_j}(\mathcal{D}))$, and that small variations in the correlation coefficients produce small changes in the persistence diagrams. This claim is justified by the fact that, if X and Y are random variables with non-null variance and X^n and Y^n denote sequences of n samples from X and Y respectively, then the sample correlation of X^n and Y^n converges in probability to the correlation between X and Y by the law of large numbers and the continuous mapping theorem [67].

In practice, $\mathcal{D}'$ is fixed to a uniform sample of $2{,}000$ elements from the original training dataset, an experimentally selected size that is large enough to obtain sufficient precision.

Sampling the functional graph. Because of computational limitations, in the case of modern DNNs less than 1% of the nodes —a priori, a statistically insignificant sample size— can be included in the persistent homology calculation. To alleviate this, we sample nodes according to a notion of importance, following ideas introduced in [68] adapted to neurons on a neural network instead of inputs of the dataset. Thus, let $\mathcal{D}'$ be some selected subsample of the training dataset. The *importance score* of a node $v \in V$ is defined as

$$I_v(\mathcal{D}') = \left| \left\{ x \in \mathcal{D}' : v = \arg\max_{w \in V} |N_w(x)| \right\} \right|, \tag{4.2}$$

where $\arg\max$ returns only one vertex in case of tie between multiple vertices —in our case, we use the tie breaking strategy implemented by the NumPy library [69]. Hence $I_v(\mathcal{D}')$ indicates

the amount of inputs from $\mathcal{D}'$ for which the activation of v is the largest (or tied-to-largest) among all nodes. Note that a majority of nodes v will have $I_v(\mathcal{D}') = 0$. This is equivalent to excluding these nodes from analysis, which is undesirable —not only because it is unclear how this will affect the application of TDA, but also because the amount of nodes with $I_v(\mathcal{D}') \neq 0$ might be low enough to severely constrain the size of a subsample. Thus, from I we construct a probability distribution P on V, artificially inflated to make sure that every element of V appears with nonzero probability. This probability $P(v)$ is defined as

$$\frac{I_v(\mathcal{D}')}{|\mathcal{D}'|+1} \text{ if } I_v(\mathcal{D}') > 0, \text{ and } \frac{1}{(|\mathcal{D}'|+1)\cdot|\{u \in V : I_u(\mathcal{D}') = 0\}|} \text{ otherwise.} \tag{4.3}$$

Specifically, we sample 3,000 nodes (without repetition) according to this probability distribution, and restrict our analysis to these nodes. This sampling is non-deterministic, and thus can be repeated a number of times to obtain n different subsamples $V_1, \ldots, V_n$. Applying the same transformations on the n resulting functional graphs we obtain n different persistence diagrams per network. Then, we use bootstrapping over the n summaries (see 3.3.3) combining them into a single one. This last representation aims to approximate the persistence summary that would be obtained without sampling.

Table 1: Top three combinations of persistence summaries per task according to their respective mean of R^2 test values in the 10 experiments of the 5×2-fold cross-validation statistical test. **ASD:** Average and standard deviation of births and deaths. **ASDSQ:** Average and standard deviation of births and deaths, concatenated with the corresponding squared values; see Section 4.1.

Task 1		
Top TDA summaries	Best dim	R^2 score
ASDSQ	0 and 1	0.5601 ± 0.13
ASDSQ	1	0.4321 ± 0.12
ASD	1	0.3720 ± 0.14
Task 2		
Top TDA summaries	Best dim	R^2 score
ASD	1	0.9337 ± 0.01
ASD	0 and 1	0.9198 ± 0.02
ASDSQ	1	0.9166 ± 0.03

4.2. Discussion

The combinations of persistence summaries that yielded the top three mean R^2 scores for the generalization gap prediction experiments are shown in Table 1. Basic statistical descriptors related to births and deaths of homology generators obtained highest scores overall, validating the results obtained in [70], in which simple vectorizations consisting of elementary statistical descriptors of persistence diagrams were the persistence summaries that obtained the best performances as input in a variety of image classification tasks. In particular, the vectors composed of averages and standard deviations of births and deaths (and their squares) were those that obtained the best R^2 scores in both tasks. Figure 4.2 shows the average performance of the entire list of summaries. These results suggest that the generalization gap is mostly linked with the average

position and dispersion of points in persistence diagrams. Summaries based on alleged predominance of larger lifetime values, such as persistence entropy or persistence pooling vectors, showed a lower predictive value.

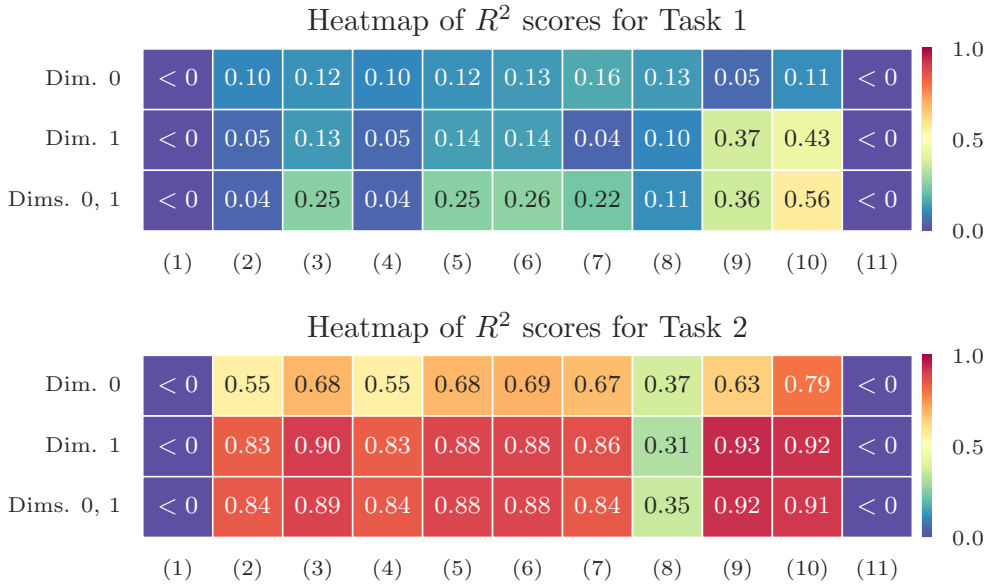

Figure 4.2: Mean R^2 test values after the 10 experiments of the 5×2-fold cross-validation statistical test for tasks 1 and 2 for the combinations of persistence summaries described in the experimental procedure of Section 4.1. Rows correspond to homological dimensions H_0, H_1, and a concatenation of both. Column numbers represent the following combinations of persistence summaries: (1) persistence pooling of 10 elements; (2) average lives and midlives; (3) average lives and midlives, original and squared; (4) average births and deaths; (5) average births and deaths, original and squared; (6) average births and deaths with a logarithmic model; (7) concatenation of combinations 3 and 5; (8) persistence entropy; (9) average and standard deviation of births and deaths; (10) average and standard deviation of births and deaths, original and squared; (11) complex polynomials with 10 coefficients.

Overall, results are more conclusive for Task 2 than for Task 1, and more significant in homological dimension 1, although some of the best R^2 scores are achieved using a combination of dimensions 0 and 1 for both tasks. It should also be noticed that R^2 scores grow when squares of summaries are added to the model, suggesting departure from linearity.

Explainability. The distribution of points in persistence diagrams is determined by correlations between neuron activation vectors. Generators of the zero-homology group H_0 of a Vietoris–Rips simplicial complex at filtration level t correspond to connected components of a functional graph in which every edge has a weight smaller than or equal to t, hence a correlation coefficient of $1-t$ in absolute value among the neurons in the group. Hence, for $t = 0$ there is one generator for each group of neurons that share correlation coefficients equal to ± 1. Points $(0, d)$ in zero-dimensional persistence diagrams arise whenever two (or more) connected components merge in the filtration at time d, and therefore they correspond to non-zero edge weights of a minimum spanning tree of the network's functional graph. High weights in a minimum spanning tree imply that the overall correlations between neurons are low. The lower the correlation between neurons,

the higher the number of nonlinearly related features learned by the neural network, and hence the stronger the real expressive power of the network. In conclusion, a combination of a high average of death values with a low standard deviation in a zero-dimensional persistence diagram is a plausible indication of an increased expressive power of the neural network, that should lead to better generalization capabilities and thus a smaller generalization gap.

Points in one-dimensional persistence diagrams correspond to cycles of the network's functional graph that are not filled by regions in the Vietoris–Rips complex. Thus a one-dimensional generator appears in the filtration at time t whenever there is a cyclically ordered group of neurons sharing correlations greater than or equal to $1 - t$ with their neighbours, which can be interpreted as a group of neurons that have learned similar features. The earlier a cycle is born, the higher the correlations among the neurons in the cycle, and the higher the death value of a cycle, the higher the differences between the features learned by non-neighbouring neurons in the cycle. Therefore, higher lifetime values may be associated with an increased number of different features learned by groups of jointly operating neurons. Thus, the higher the deaths in the one-dimensional persistence diagram of the functional graph of a neural network, the more expressive power the neural network may have, and thus the better it may generalize.

Clustering. The interpretations described in the previous subsection are consistent with what is shown in Figure 4.3. In this figure, each row represents a different task, each column represents a different persistent summary, and each point in a cell corresponds to a neural network for the given task. The two rows, upper and lower, represent Task 1 and Task 2, respectively. The first and third columns represent the average of deaths of zero- and one-dimensional persistence diagrams, whereas the second and fourth columns represent the standard deviation of deaths of zero- and one-dimensional persistence diagrams, respectively. In the first and second rows, neural networks are clustered according to the number of convolutional blocks and the number of convolutional layers that each network contains.

Figure 4.3 suggests that persistence summaries detect very neatly the clusters of neural networks in each task. Naturally, the generalization gap is strongly influenced by the depth of the networks, which is almost determined by the number of convolution blocks and layers. When the number of convolutions is fixed, we see a consistent behavior: the higher the average deaths and the lower the standard deviations, the better the network's performance. This discovery has the potential of being used for network regularization. Support for this assertion was provided in [9] through the successful implementation of regularizers designed to raise the average number of deaths while reducing standard deviations.

We further analyzed if persistence diagrams for individual labels in a classification task were different between them, in order to gain insight about what was influencing TDA methods and functional graphs the most. We computed persistence diagrams in dimensions 0 and 1 per different neural network and per label. The datasets used to recreate functional graphs were restrictions of the test set to each label. Details and figures can be found in the Appendix. Similar results were seen when comparing these persistence diagrams with the original ones. The majority of class-dependent persistence diagrams whose DNNs obtained extreme accuracies, i.e., highest and lowest, were analogous to the diagrams in the class-independent case. This shows that functional graphs are robust to unbalanced datasets in terms of the number of samples per label.

Persistence summaries. Results show that linear models of persistence summaries can predict the generalization gap, since we obtained competitive results in both tasks, as seen in Table 1 and Table 2. However, the fact that a summary based on a combination of non-linear transformations of persistence features yielded the best score for Task 1 suggests that more complex models can

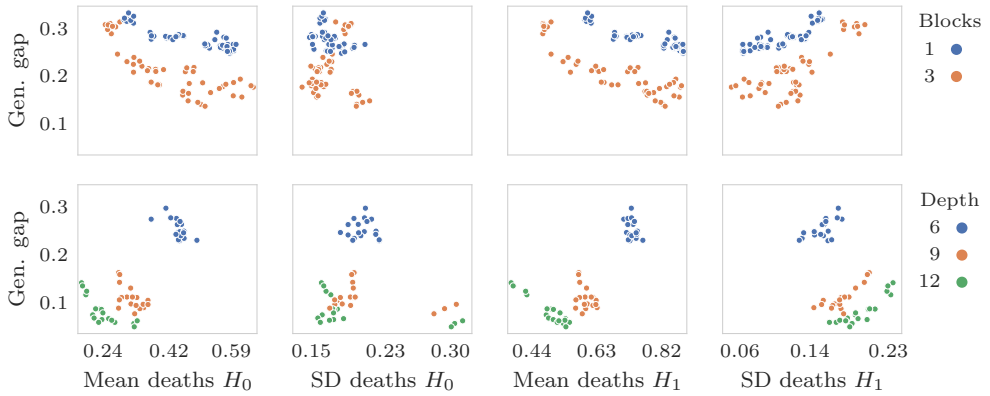

Figure 4.3: Averages and standard deviations of deaths for persistence diagrams in dimension 0 (first two columns) and dimension 1 (last two columns) for Task 1 (first row) and Task 2 (second row). For Task 1, points represent 96 VGG-like neural networks trained on the CIFAR10 dataset; blue and orange points represent neural networks with one and three convolutional blocks, respectively. For Task 2, points correspond to 54 *network in network* architectures trained on the SVHN dataset; blue, orange, and green points represent neural networks with six, nine, and twelve convolutional layers, respectively.

Table 2: Comparison of our best performing summaries with state of the art: Average and standard deviation of R^2 scores for Task 1 and Task 2 computed from linear models trained in the ten cases of the 5×2-fold cross-validation.

	Task 1	Task 2
Interpex	-0.0518 ± 0.06	0.9500 ± 0.01
Always Generalize	0.9715 ± 0.01	0.8893 ± 0.02
BrAIn	0.4520 ± 0.08	0.7180 ± 0.04
Ours	0.5601 ± 0.13	0.9337 ± 0.01

have better capacity to relate persistence summaries to the generalization gap.

When it comes to ranking summaries, persistence pooling and complex polynomials produced the lowest R^2 scores overall, as shown in Fig. 4.2. For persistence pooling, one possible explanation of its low performance is that it relies on lifetimes of points that live the longest, in contrast to the most effective summaries, which are based on average location and dispersion of the whole set of points in a persistence diagram. Similarly, truncated complex polynomials are not sufficiently accurate measures of the location and aggregation of the collection of all points in persistence diagrams. The fact that persistence entropy achieves non-optimal R^2 scores for Task 2 in Fig. 4.2 is consistent with the interpretation that the distribution of points near the diagonal in one-dimensional persistence diagrams is substantial for generalization gap prediction.

State-of-the-art comparison. Table 2 shows a comparison of the results of our best performing linear models based on persistence summaries with state-of-the-art methods. In this table, the R^2 scores describe the ability of each linear model to predict the generalization gap with respect to the coefficient of determination. Table 3 shows the pairwise p-values between the linear models induced by our best performing combinations of persistence summaries, shown in Table 1, and the linear models induced by the winning generalization measures of the PGDL dataset.

Table 3: Statistical p-values of the pairwise 5×2-fold cross-validation significance test proposed in [62], with the coefficient of determination R^2 as performance metric. The null hypothesis is that, for a fixed-size random drawn training dataset, the two linear models trained with our combinations of persistence summaries or with the winning generalization measures of the PGDL competition have the same R^2 score on a randomly drawn test dataset. Boldface p-values are lower than 0.05. **ASD1:** Average and standard deviation of births and deaths of dimension one. **ASDSQ01:** Average and standard deviation of births and deaths, concatenated with their squared values, for dimensions zero and one. Their R^2 scores are shown in Table 1.

Task 1	Interpex	Always Generalize	BrAIn	ASDSQ01
ASDSQ01	**0.00**	**0.01**	0.23	
ASD1	**0.03**	**0.00**	0.55	0.13
Task 2				
ASDSQ01	0.37	0.80	0.37	
ASD1	0.19	**0.00**	**0.01**	0.51

We obtain the second-best mean R^2 scores for both tasks, after Always Generalize and Interpex in the first and second ones, respectively. However, assuming that two methods are significantly different whenever their pairwise p-value is lower than 0.05 in Table 3, there is no significant difference between the R^2 scores of the linear models of our best combination of persistence summaries in Task 2 and the linear models induced by the generalization measure of the Interpex team. Additionally, our models are significantly better than those for the Interpex generalization measure in Task 1 and than the ones for the generalization measures of Always Generalize and BrAIn in Task 2. These results suggest that persistence summaries are a promising tool to develop robust models to predict the generalization gap.

4.3. Hardware, software and licenses

Persistence diagrams were computed with Python `giotto-ph` [71] (GNU AGPLv3) using a Quadro P6000 GPU. Persistence summaries were computed with the `giotto-tda` framework [72] (AGPLv3 License), and density curves were drawn using SciPy 1.8.0 [73]. Analysis was performed on a personal computer with an Intel Core i7 (4th generation) processor with an NVIDIA GeForce GTX 960M 2GB GDDR5, using the libraries Jupyter Notebook (New BSD License), NumPy (BSD 3-Clause "New" or "Revised" License) and TensorFlow with Keras (Apache 2.0 License). Docker (Apache 2.0 License) was also used to perform the experiments. The dataset of neural networks from [4] is licensed under Apache 2.0.

5. Conclusions

We have defined a framework that can be used to explore interpretability of DNNs based on topological properties of their functional graphs. This relaxes the problem of understanding the internal representations of a neural network to, in a broad sense, understanding their *shape*. Regarding generalization, we have shown examples of how one can interpret DNN neuron interactions based on their correlations by means of persistence diagrams. Moreover, we proved that the generalization gap can be consistently predicted using topological persistence summaries extracted from functional graphs, with a competitive prediction accuracy on two different computer vision problems. The most successful summaries were those related with the average location

and dispersion of points in persistence diagrams. Hence, it is not true in our case that points near the diagonal in persistence diagrams are irrelevant, as often claimed in TDA studies.

Limitations. A practical limitation of persistent homology comes from its computational complexity —sampling methods are not necessarily optimal and information might be lost in sampling processes for datasets and for neurons. Transformations of persistence diagrams into summaries may also cause a loss of information; however, this seems unavoidable if one wants to obtain easy-to-compute generalization measures.

Future work. Although we found strong patterns relating persistence summaries with generalization gaps (Figure 4.3), broader experimentation is required to see if these patterns are consistent among other kinds of networks and machine learning tasks, and also to make more explicit which features of the networks are involved in the TDA-driven clustering effect that we have observed.

The mere definition of functional graphs raises a question: which is the optimal metric to compare neurons given an architecture? There might be better alternatives to linear correlation between activation vectors; for instance, Spearman correlation was used in co-activation graphs for a similar purpose in [74].

Another problem is to find an optimal neuron sampling strategy. This is related to the problem of finding the most relevant neurons in a DNN graph. Persistence summaries suggest that grouping neurons in terms of their activation structure is feasible for DNNs. However, understanding which functional phenomena are being captured into such communities of nodes needs further study. This could lead to the discovery of new architectural properties useful to develop better networks.

Figure 4.3 shows that, fixing the depth of a neural network, there is a consistent association between a lower generalization gap and a higher average of death values together with a small dispersion in the persistence diagrams of the network's functional graph in dimensions zero and one. This finding has the potential to improve the performance of a given architecture during training by means of a regularization term that maximizes averages of deaths while minimizing standard deviation, using the framework for differential calculus on persistence diagrams discussed in [75, 76]. This work plan was undertaken in [9].

References

[1] A. Azulay, Y. Weiss, Why do deep convolutional networks generalize so poorly to small image transformations?, Journal of Machine Learning Research 20 (184) (2019) 1–25.
URL http://jmlr.org/papers/v20/19-519.html 2

[2] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6572 2

[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Communications of the ACM 64 (2021) 107–115. doi:https://doi.org/10.1145/3446776. 2

[4] Y. Jiang, P. Foret, S. Yak, D. M. Roy, H. Mobahi, G. K. Dziugaite, S. Bengio, S. Gunasekar, I. Guyon, B. Neyshabur, NeurIPS 2020 Competition: Predicting generalization in deep learning (2020). arXiv:2012.07976. 2, 3, 4, 7, 15

[5] C. Lassance, L. Béthune, M. Bontonou, M. Hamidouche, V. Gripon, Ranking deep learning generalization using label variation in latent geometry graphs (2020). arXiv:2011.12737. 2, 3

[6] P. Natekar, M. Sharma, Representation based complexity measures for predicting generalization in deep learning (2020). arXiv:2012.02775. 2, 3

[7] S. Aithal K, D. Kashyap, N. Subramanyam, Robustness to augmentations as a generalization metric (2021). arXiv:2101.06459. 2, 3
[8] H. Edelsbrunner, J. Harer, Computational Topology: An Introduction, American Mathematical Society, 2010. doi:https://doi.org/10.1090/mbk/069. 2, 6
[9] R. Ballester, C. Casacuberta, S. Escalera, Decorrelating neurons using persistence, in: NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations, Proceedings of Machine Learning Research, 2023.
URL openreview.net/forum?id=NJS6568y79 2, 13, 16
[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Commun. ACM 64 (3) (2021) 107–115. doi:10.1145/3446776.
URL https://doi.org/10.1145/3446776 3
[11] G. K. Dziugaite, D. M. Roy, Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, in: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Sydney, 2017.
URL https://arxiv.org/abs/1703.11008 3
[12] V. Nagarajan, J. Z. Kolter, Generalization in deep networks: The role of distance from initialization, in: Neural Information Processing Systems (NeurIPS) – Deep Learning: Bridging Theory and Practice, 2017.
URL https://arxiv.org/abs/1901.01672 3
[13] K. Kawaguchi, Y. Bengio, L. Kaelbling, Generalization in deep learning, in: Mathematical Aspects of Deep Learning, Cambridge University Press, 2022, pp. 112–148. doi:10.1017/9781009025096.003.
URL https://doi.org/10.1017%2F9781009025096.003 3
[14] P. L. Bartlett, D. J. Foster, M. J. Telgarsky, Spectrally-normalized margin bounds for neural networks, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://proceedings.neurips.cc/paper_files/paper/2017/file/b22b257ad0519d4500539da3c8bcf4dd-Paper.pdf 3
[15] N. Golowich, A. Rakhlin, O. Shamir, Size-independent sample complexity of neural networks, in: S. Bubeck, V. Perchet, P. Rigollet (Eds.), Proceedings of the 31st Conference On Learning Theory, Vol. 75 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 297–299.
URL https://proceedings.mlr.press/v75/golowich18a.html 3
[16] T. Liang, T. Poggio, A. Rakhlin, J. Stokes, Fisher-Rao metric, geometry, and complexity of neural networks, in: K. Chaudhuri, M. Sugiyama (Eds.), Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Vol. 89 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 888–896.
URL https://proceedings.mlr.press/v89/liang19a.html 3
[17] B. Dupuis, G. Deligiannidis, U. Simsekli, Generalization bounds using data-dependent fractal dimensions, in: International Conference on Machine Learning, Vol. 40, 2023. 3
[18] S. Lotfi, M. A. Finzi, S. Kapoor, A. Potapczynski, M. Goldblum, A. G. Wilson, PAC-Bayes compression bounds so tight that they can explain generalization, in: A. H. Oh, A. Agarwal, D. Belgrave, K. Cho (Eds.), Advances in Neural Information Processing Systems, 2022.
URL https://openreview.net/forum?id=o8nYuR8ekFm 3
[19] U. Simsekli, O. Sener, G. Deligiannidis, M. A. Erdogdu, Hausdorff dimension, heavy tails, and generalization in neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 5138–5151.
URL https://proceedings.neurips.cc/paper_files/paper/2020/file/37693cfc748049e45d87b8c7d8b9aacd-Paper.pdf 3
[20] M. Hardt, B. Recht, Y. Singer, Train faster, generalize better: Stability of stochastic gradient descent, in: M. F. Balcan, K. Q. Weinberger (Eds.), Proceedings of The 33rd International Conference on Machine Learning, Vol. 48 of Proceedings of Machine Learning Research, PMLR, New York, New York, USA, 2016, pp. 1225–1234.
URL https://proceedings.mlr.press/v48/hardt16.html 3
[21] B. Neyshabur, S. Bhojanapalli, D. Mcallester, N. Srebro, Exploring generalization in deep learning, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://proceedings.neurips.cc/paper_files/paper/2017/file/10ce03a1ed01077e3e289f3e53c72813-Paper.pdf 3
[22] Y. Jiang, D. Krishnan, H. Mobahi, S. Bengio, Predicting the generalization gap in deep networks with margin distributions, ICLR, 2019. arXiv:1810.00113. 3
[23] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht, The marginal value of adaptive gradient methods in machine learning, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 4151–4161.
URL https://dl.acm.org/doi/pdf/10.5555/3294996.3295170 3
[24] P. Chaudhari, S. Soatto, Stochastic gradient descent performs variational inference, converges to limit cycles for

deep networks, in: 2018 Information Theory and Applications Workshop (ITA), 2018, pp. 1–10. doi:10.1109/ITA.2018.8503224. 3

[25] S. L. Smith, Q. V. Le, A Bayesian perspective on generalization and stochastic gradient descent, in: International Conference on Learning Representations (ICLR), 2018.
URL https://openreview.net/pdf?id=BJij4yg0Z 3

[26] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, S. Bengio, Fantastic generalization measures and where to find them, ICLR, 2020.
URL https://openreview.net/pdf?id=SJgIPJBFvH 3

[27] Y. Jiang, P. Natekar, M. Sharma, S. K. Aithal, D. Kashyap, N. Subramanyam, C. Lassance, D. M. Roy, G. K. Dziugaite, S. Gunasekar, I. Guyon, P. Foret, S. Yak, H. Mobahi, B. Neyshabur, S. Bengio, Methods and analysis of the first competition in predicting generalization of deep learning, in: H. J. Escalante, K. Hofmann (Eds.), Proceedings of the NeurIPS 2020 Competition and Demonstration Track, Vol. 133 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 170–190.
URL https://proceedings.mlr.press/v133/jiang21a.html 3

[28] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1 (2) (1979) 224–227. doi:10.1109/TPAMI.1979.4766909. 3

[29] Y. Schiff, B. Quanz, P. Das, P.-Y. Chen, Predicting deep neural network generalization with perturbation response curves, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 21176–21188.
URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b0dd033cbe58aa5ea27747271bfd84e3-Paper.pdf 3

[30] V. Narayanaswamy, R. Anirudh, I. Kim, Y. Mubarka, A. Spanias, J. J. Thiagarajan, Predicting the generalization gap in deep models using anchoring, in: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 4393–4397. doi:10.1109/ICASSP43922.2022.9747136. 3

[31] F. Hensel, M. Moor, B. Rieck, A survey of topological machine learning methods, Frontiers in Artificial Intelligence 4 (2021). doi:10.3389/frai.2021.681108.
URL https://www.frontiersin.org/articles/10.3389/frai.2021.681108 3

[32] S. Chowdhury, T. Gebhart, S. Huntsman, M. Yutin, Path homologies of deep feedforward networks, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1077–1082. doi:10.1109/ICMLA.2019.00181. 3

[33] M. Bianchini, F. Scarselli, On the complexity of neural network classifiers: A comparison between shallow and deep architectures, IEEE Transactions on Neural Networks and Learning Systems 25 (8) (2014) 1553–1565. doi:10.1109/TNNLS.2013.2293637. 3

[34] W. H. Guss, R. Salakhutdinov, On characterizing the capacity of neural networks using algebraic topology, arXiv preprint arXiv:1802.04443 (2018). 3

[35] K. N. Ramamurthy, K. Varshney, K. Mody, Topological data analysis of decision boundaries with application to model selection, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 5351–5360.
URL https://proceedings.mlr.press/v97/ramamurthy19a.html 3

[36] W. Li, G. Dasarathy, K. Natesan Ramamurthy, V. Berisha, Finding the homology of decision boundaries with active learning, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 8355–8365.
URL https://proceedings.neurips.cc/paper_files/paper/2020/file/5f14615696649541a025d3d0f8e0447f-Paper.pdf 3

[37] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, S. Soatto, Empirical study of the topology and geometry of deep networks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 3762–3770. doi:10.1109/CVPR.2018.00396. 3

[38] B. Liu, M. Shen, Some geometrical and topological properties of DNNs' decision boundaries, Theor. Comput. Sci. 908 (C) (2022) 64–75. doi:10.1016/j.tcs.2021.11.013.
URL https://doi.org/10.1016/j.tcs.2021.11.013 3

[39] G. Petri, A. Leitão, On the topological expressive power of neural networks, in: TDA & Beyond, 2020.
URL https://openreview.net/forum?id=I44kJPuvqPD 3

[40] V. Khrulkov, I. Oseledets, Geometry score: A method for comparing generative adversarial networks, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 2621–2629.
URL https://proceedings.mlr.press/v80/khrulkov18a.html 3

[41] S. Zhou, E. Zelikman, F. Lu, A. Y. Ng, G. E. Carlsson, S. Ermon, Evaluating the disentanglement of deep generative models through manifold topology, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=djwS0m4Ft_A 3

[42] G. Carlsson, R. B. Gabrielsson, Topological approaches to deep learning, in: N. A. Baas, G. E. Carlsson, G. Quick, M. Szymik, M. Thaule (Eds.), Topological Data Analysis, Springer International Publishing, Cham, 2020, pp. 119–146. 3

[43] R. B. Gabrielsson, G. Carlsson, Exposition and interpretation of the topology of neural networks (2018). doi:10.48550/ARXIV.1810.03234.
URL https://arxiv.org/abs/1810.03234 3

[44] B. Rieck, M. Togninalli, C. Bock, M. Moor, M. Horn, T. Gumbsch, K. Borgwardt, Neural persistence: A complexity measure for deep neural networks using algebraic topology, in: International Conference on Learning Representations, 2019.
URL https://openreview.net/forum?id=ByxkijC5FQ 3

[45] T. Gebhart, P. Schrater, A. Hylton, Characterizing the shape of activation space in deep neural networks, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1537–1542. doi:10.1109/ICMLA.2019.00254. 3

[46] L. Girrbach, A. Christensen, O. Winther, Z. Akata, A. S. Koepke, Caveats of neural persistence in deep neural networks, in: 2nd Annual TAG in Machine Learning, 2023. 3

[47] T. Birdal, A. Lou, L. Guibas, U. Simsekli, Intrinsic dimension, persistent homology and generalization in neural networks, in: A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, 2021.
URL https://openreview.net/forum?id=099uYP0EKsJ 3, 4

[48] C. A. Corneanu, M. Madadi, S. Escalera, A. M. Martinez, What does it mean to learn in deep networks? And, how does one detect adversarial attacks?, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4752–4761. doi:10.1109/CVPR.2019.00489. 3, 4

[49] C. A. Corneanu, S. Escalera, A. M. Martinez, Computing the testing error without a testing set, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 2674–2682. doi:10.1109/CVPR42600.2020.00275. 3, 4, 7

[50] G. Kozma, Z. Lotker, G. Stupp, The minimal spanning tree and the upper box dimension, Proceedings of the American Mathematical Society 134 (4) (2006) 1183–1187. 3

[51] B. Schweinhart, Persistent homology and the upper box dimension, Discrete & Computational Geometry 65 (2) (2021) 331–364. doi:10.1007/s00454-019-00145-3.
URL https://doi.org/10.1007/s00454-019-00145-3 3

[52] F. Chazal, V. de Silva, S. Oudot, Persistence stability for geometric complexes, Geometriae Dedicata 173 (2014) 193–214. doi:https://doi.org/10.1007/s10711-013-9937-z. 6

[53] S. Chowdhury, F. Mémoli, A functorial Dowker theorem and persistent homology of asymmetric networks, Journal of Applied and Computational Topology 173 (2) (2018) 115–175. doi:https://doi.org/10.1007/s41468-018-0020-6. 6

[54] M. Mézard, A. Montanari, Information, Physics, and Computation, Oxford University Press, 2009. doi:DOI:10.1093/acprof:oso/9780198570837.001.0001. 7

[55] N. Atienza, R. Gonzalez-Diaz, M. Soriano-Trigueros, On the stability of persistent entropy and new summary functions for topological data analysis, Pattern Recognition 107 (2020) 107509. doi:https://doi.org/10.1016/j.patcog.2020.107509. 7

[56] T. Bonis, M. Ovsjanikov, S. Oudot, F. Chazal, Persistence-based pooling for shape pose recognition, in: Bac, A., Mari, J. L. (Eds.), Computational Topology in Image Context, CTIC 2016, Lecture Notes in Computer Science, Vol. 9667, Springer, Cham, 2016, pp. 19–29. doi:https://doi.org/10.1007/978-3-319-39441-1_3. 7

[57] B. D. Fabio, M. Ferri, Comparing persistence diagrams through complex vectors, in: V. Murino and E. Puppo (Eds.), Image Analysis and Processing – ICIAP 2015, Vol. 9279 of Lecture Notes in Computer Science, Springer, 2015. doi:https://doi.org/10.1007/978-3-319-23231-7_27. 7

[58] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: The 3rd International Conference on Learning Representations (ICLR2015), 2015.
URL https://arxiv.org/abs/1409.1556 8

[59] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep. (2009).
URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf 8, 10

[60] M. Lin, Q. Chen, S. Yan, Network in network (2014). arXiv:1312.4400. 8

[61] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf 8

[62] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (7) (1998) 1895–1923. doi:10.1162/089976698300017197.
URL https://doi.org/10.1162/089976698300017197 9, 15

[63] Y. Zhou, H. Chang, Y. Lu, X. Lu, R. Zhou, Improving the performance of VGG through different granularity feature combinations, IEEE Access 9 (2020) 26208–26220. 10
[64] Edelsbrunner, Letscher, Zomorodian, Topological persistence and simplification, Discrete & Computational Geometry 28 (4) (2002) 511–533. doi:10.1007/s00454-002-2885-2.
URL https://doi.org/10.1007/s00454-002-2885-2 10
[65] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009. 10
[66] N. Milosavljević, D. Morozov, P. Škraba, Zigzag persistent homology in matrix multiplication time, in: Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, SoCG'11, Association for Computing Machinery, New York, NY, USA, 2011, p. 216–225.
URL https://www.mrzv.org/publications/zzph-mmt/socg11/ 10
[67] A. W. van der Vaart, Asymptotic Statistics, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press. doi:10.1017/CBO9780511802256. 10
[68] E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, J. Luo, Adaptive hierarchical down-sampling for point cloud classification, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 12953–12961. doi:10.1109/CVPR42600.2020.01297. 10
[69] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362. doi:10.1038/s41586-020-2649-2.
URL https://doi.org/10.1038/s41586-020-2649-2 10
[70] D. Ali, A. Asaad, M.-J. Jimenez, V. Nanda, E. Paluzo-Hidalgo, M. Soriano-Trigueros, A survey of vectorization methods in topological data analysis (2022). arXiv:2212.09703. 11
[71] J. B. Pérez, S. Hauke, U. Lupo, M. Caorsi, A. Dassatti, Giotto-ph: A Python library for high-performance computation of persistent homology of Vietoris–Rips filtrations (2021). arXiv:2107.05412. 15
[72] G. Tauzin, U. Lupo, L. Tunstall, J. B. Pérez, M. Caorsi, A. M. Medina-Mardones, A. Dassatti, K. Hess, Giotto-tda: A topological data analysis toolkit for machine learning and data exploration, Journal of Machine Learning Research 22 (39) (2021) 1–6.
URL http://jmlr.org/papers/v22/20-325.html 15
[73] P. Virtanen et al., SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nature Methods 17 (2020) 261–272. doi:https://doi.org/10.1038/s41592-019-0686-2. 15
[74] V. A. C. Horta, I. Tiddi, S. Little, A. Mileo, Extracting knowledge from Deep Neural Networks through graph analysis, Future Generation Computer Systems 120 (2021) 109–118. doi:https://doi.org/10.1016/j.future.2021.02.009. 16
[75] J. Leygonie, S. Oudot, U. Tillmann, A framework for differential calculus on persistence barcodes, Foundations of Computational Mathematics 22 (4) (2022) 1069–1131. doi:10.1007/s10208-021-09522-y.
URL https://doi.org/10.1007/s10208-021-09522-y 16
[76] M. Carrière, F. Chazal, M. Glisse, Y. Ike, H. Kannan, Y. Umeda, Optimizing persistent homology based functions, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 1294–1303.
URL https://proceedings.mlr.press/v139/carriere21a.html 16

Appendix

This appendix contains an analysis per label of persistence diagrams in dimensions 0 and 1. The datasets that we used to recreate functional graphs were the restriction of the test sets to each label. We computed accuracy for each of these test subsets, and plotted persistence diagrams corresponding to those neural networks that achieved the maximum and minimum accuracies on test subsets per label for dimensions 0 and 1. The results can be seen in Figures 5.1, 5.2, 5.3 and 5.4. These results are consistent with what we found in persistence diagrams computed with the whole training dataset. Thus we see that distinction between inputs of different labels does not have a substantial influence on the distribution of points in persistence diagrams.

For a more convenient visualization, persistence diagrams in dimension 0 have been replaced with lifetime density curves, calculated by means of Gaussian kernels. Lifetime values are equal

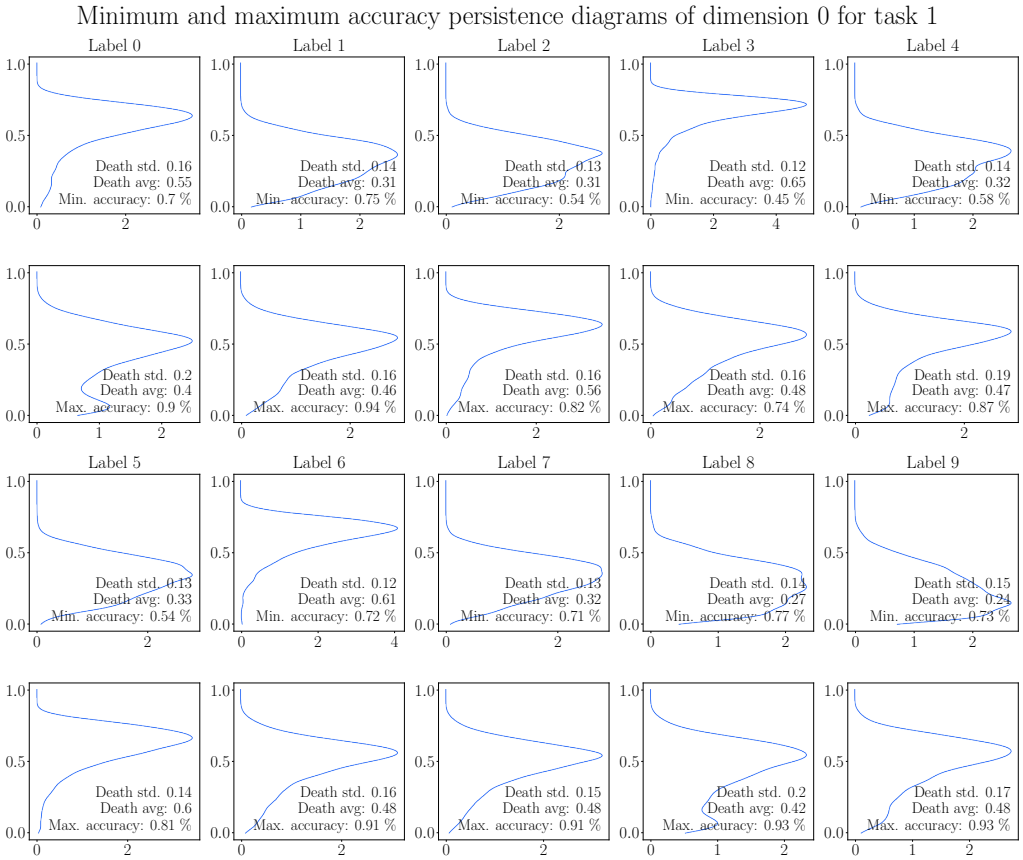

Figure 5.1: Lifetime densities in persistence diagrams in homological dimension zero of 96 VGG-like neural networks with minimum and maximum accuracies on the test set per label for Task 1.

to death values for zero-homology generators.

It can be seen in Fig. 5.3 and Fig. 5.4 that increased accuracy values for Task 2 match with scattering of points downwards the diagonal of the persistence diagram in dimension 1 and with a lower average life in dimension 0. This pattern is apparently not consistent with other architectures, such as those used in Task 1. This is explained by the splitting of network types into clusters as observed in Fig. 4.3, since for Task 2 the regression line for average deaths has negative slope in each cluster, while it has positive slope if clustering is not taken into account.

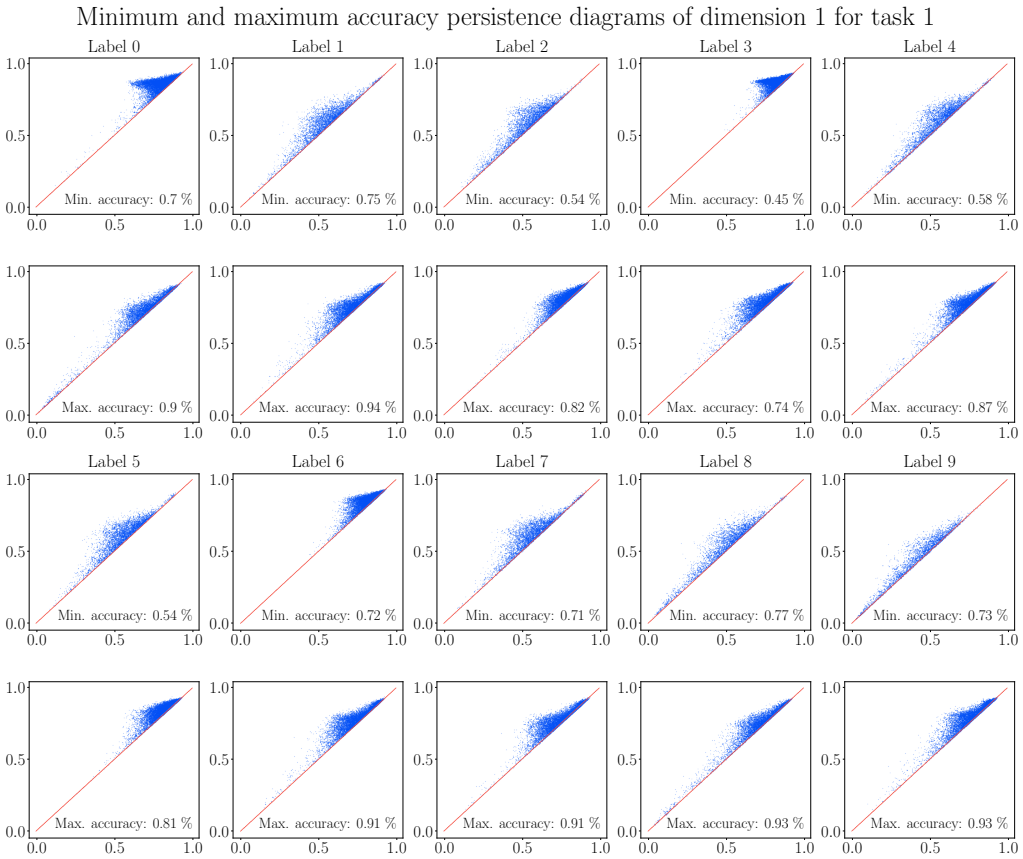

Figure 5.2: Persistence diagrams in homological dimension one of 96 VGG-like neural networks with minimum and maximum accuracies on the test set per label for Task 1.

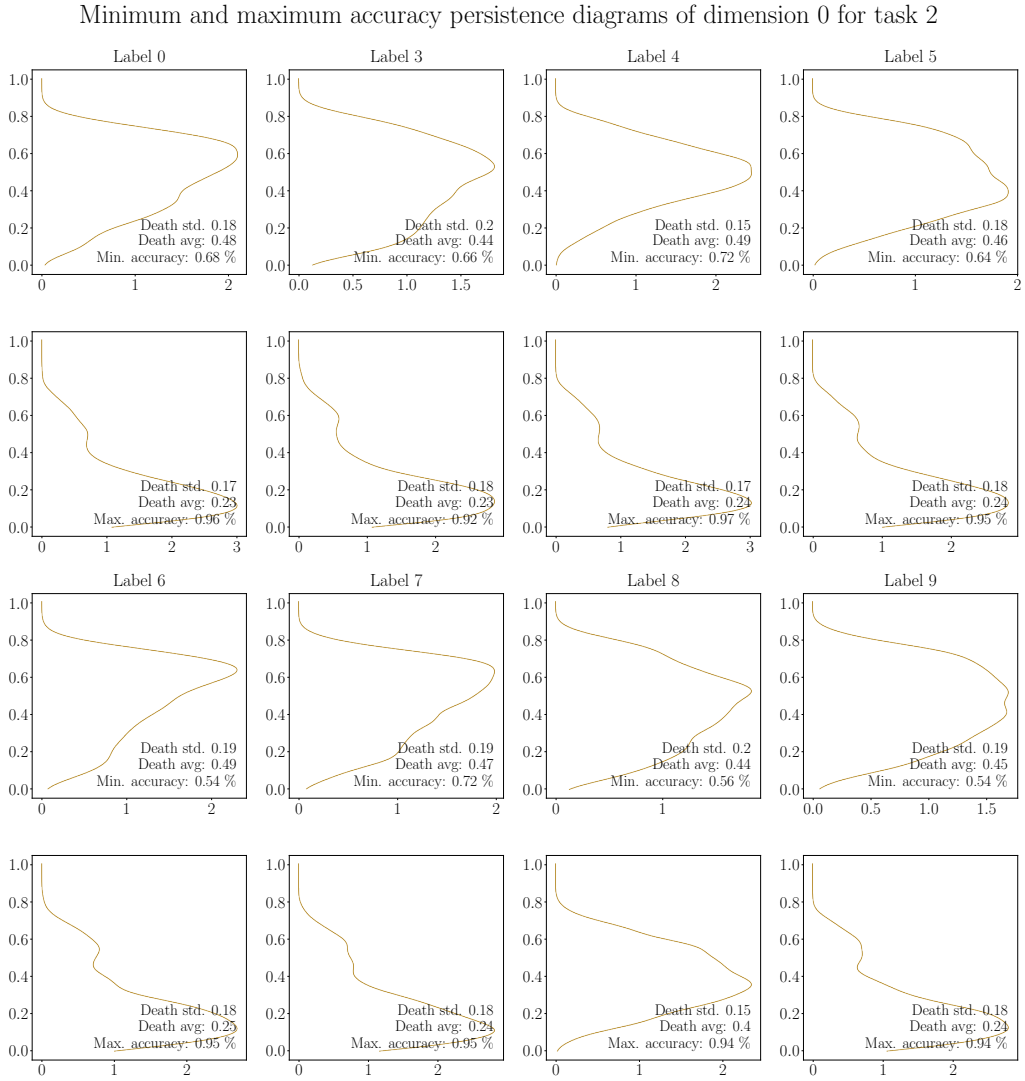

Figure 5.3: Lifetime densities in persistence diagrams in homological dimension zero of 54 *network in network* architectures with minimum and maximum accuracies on the test set per label for Task 2.

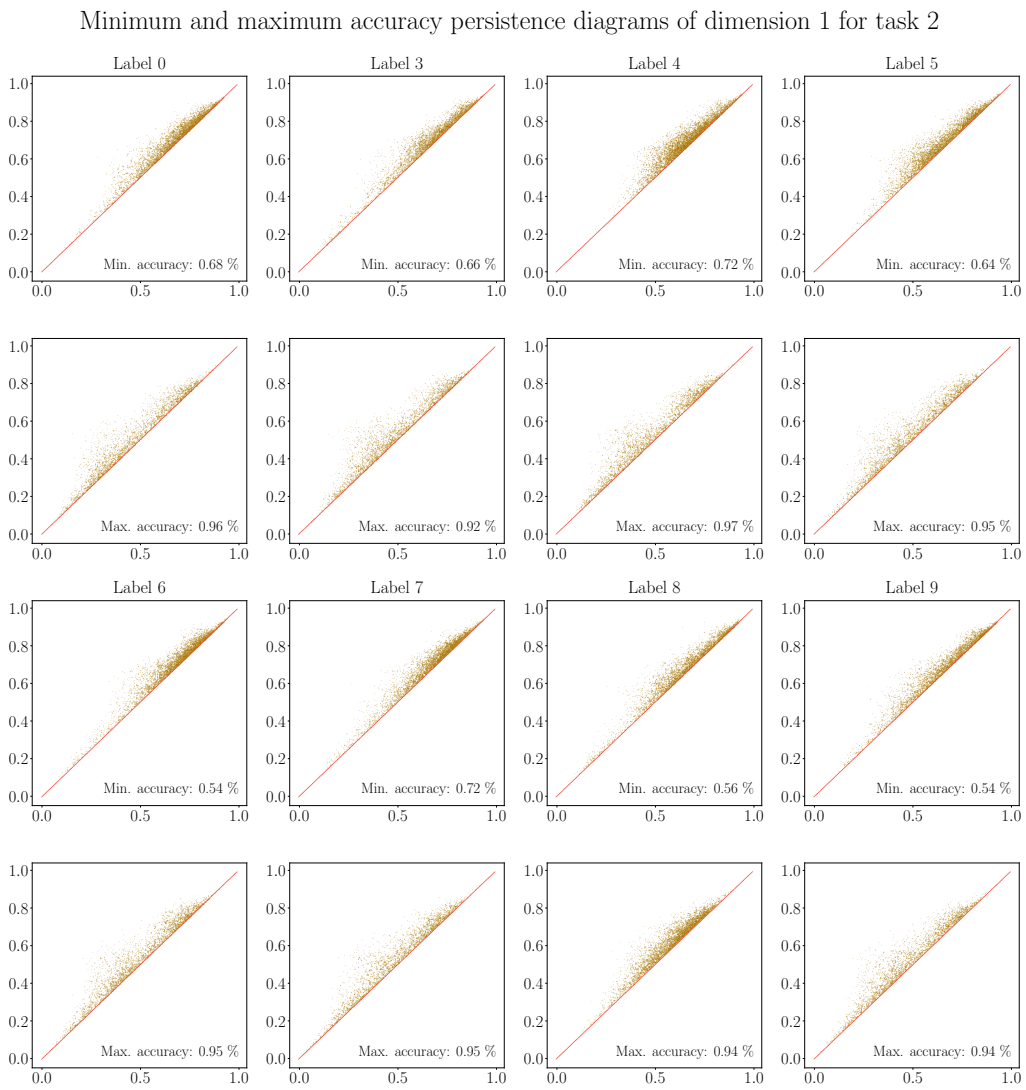

Figure 5.4: Persistence diagrams in homological dimension one of 54 *network in network* architectures with minimum and maximum accuracies on the test set per label for Task 2.