Persistent Homology Captures the Generalization of Neural Networks Without A Validation Set

Anonymous authors
Paper under double-blind review

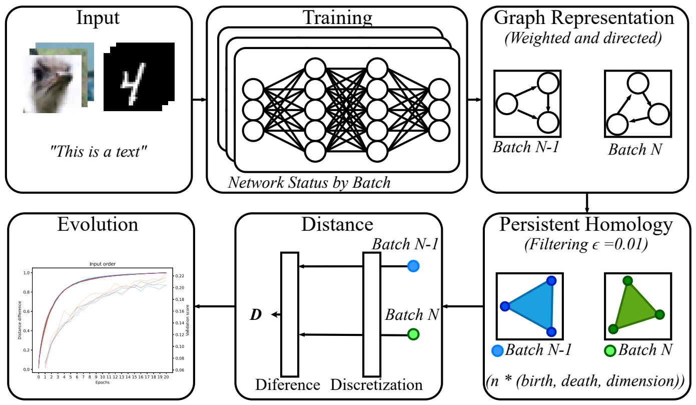


Figure 1: Our proposal.

Abstract

The training of neural networks is usually monitored with a validation (holdout) set to estimate the generalization of the model. This is done instead of measuring intrinsic properties of the model to determine whether it is learning appropriately. In this work, we suggest studying the training of neural networks with Algebraic Topology, specifically Persistent Homology (PH). Using simplicial complex representations of neural networks, we study the PH diagram distance evolution on the neural network learning process with different architectures and several datasets. Results show that the PH diagram distance between consecutive neural network states correlates with the validation accuracy, implying that the generalization error of a neural network could be intrinsically estimated without any holdout set.

1 Introduction

Generalization is what makes a machine learning model useful; the uncertainty of its behaviour with unseen data is what makes it potentially dangerous. Thus, understanding the generalization error of a model can be considered one of the holy grails of the entire machine learning field.

Machine learning practitioners typically monitor some metrics of the model to estimate its generalization error and stop the training even before the numerical convergence to prevent the overfitting of the model. Usually, the error measure or the metric relevant to the task is computed for a holdout set, the validation set. Since these data have not been directly used for updating the parameters, it is assumed that the performance of the model on the validation set can be used as a proxy of the generalization error, provided it is representative of the data that will be used in inference. One can, though, potentially overfit to this holdout set if is repeatedly used for guiding a hyperparameter search.

Instead of relying on an external set, though, the question of whether it could be possible to estimate the generalization error with some intrinsic property of the model is highly relevant, and it has been barely explored in the literature. On the other hand, Algebraic Topology has recently been gaining momentum as a mathematical tool for studying graphs, machine learning algorithms, and data.

In this work, we have the goal of, once having characterized neural networks as weighted, acyclic graphs, represented as Algebraic Topology objects (following previous works), computing distances between consecutive neural network states. More specifically, we can calculate the Persistent Homology (PH) diagram distances between a give state (i.e., when having a specific weights during the training process) and the next one (i.e., after having updated the weights in a training step), as depicted in Figure 1. We observe that during the training procedure of neural networks we can measure this distance in each learning step, and show that there exists a high correlation with the corresponding validation accuracy of the model. We do so in a diverse set of deep learning benchmarks and model hyperparameters. This shines light on the question of whether the generalization error could be estimated from intrinsic properties of the model, and opens the path towards a better theoretical understanding of the dynamics of the training of neural networks.

In summary, our contributions are as follows:

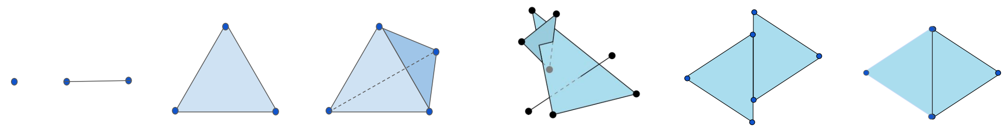
- Based on principles of Algebraic Topology, we propose measuring the distances (Silhouette and Heat) between the PH persistence diagrams obtained from a given state of a neural network during the training procedure and the one in the immediately previous weights update.
- We empirically show that the evolution of these measures during training correlate with the accuracy in the validation set. We do so in diverse benchmarks (MNIST, CIFAR10, CIFAR100, Reuters text classification), and models (MLPs in MNIST and Reuters, MLPs and CNNs in CIFAR100 and CIFAR100).
- We thus provide empirical proof of the fact that valuable information related to the learning process of neural networks can be obtained from PH distances between persistence diagrams (we will call this process *homological convergence*). In particular, we show that homological convergence is related to learning process and the generalization properties of neural networks.
- In practice, we provide a new tool for monitoring the training of neural networks, and open the path to estimating their generalization error without a validation set.

The remainder of this article is as follows. In Section 2 we describe the theoretical background of our proposal in terms of Algebraic Topology, while in Section 3 we go through the related work. Then, in Section 4 we formalize our method. Finally, in sections 6 and 7 we present and discuss our empirical results, respectively.

2 Background

In this section we introduce the mathematical foundations of this paper. A detailed mathematical description is included in the Supplementary Material.

A simplicial complex is a set composed of points, line segments, triangles, and their n-dimensional counterparts, named simplex (K). In particular, a simplicial complex must comply with two properties: 1. Every face of a simplex is also in the simplicial complex (of lower dimension). 2. The non-empty intersection of any two simplices contained on a simplicial complex is a face of both. 0,1,2,3-simplex and non simplex examples are shown in Figure 2.



(a) 0,1,2,3-simplex

(b) Non-simplex

Figure 2: Simplex and non-simplex examples.

We can associate to an undirected graph, $G = (V, E)$, a simplicial complex where all the vertices of G are the 0-simplex of the simplicial complex and the complete subgraphs with i vertices, in G

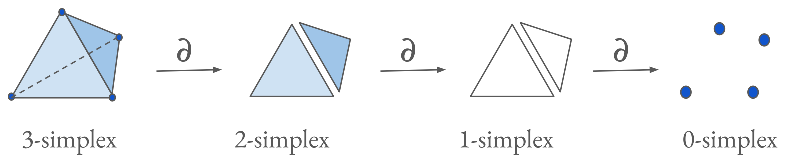


Figure 4: Boundary function sample.

corresponds to a $(i-1)$-simplex. This type of construction is usually called a complex clique on the graph G, and is denoted by $Cl(G)$. Figure 3 shows a graph clique complex Cl(G) example.

The boundary function is defined as a map, from an i-simplex to an $(i-1)$-simplex, as the sum of its $(i-1)$-dimensional faces. A boundary function sample is shown in Figure 4.

In algebraic topology, a k-chain is a combination of k-simplices (sometimes symbolized as a linear combination of simplices that compose the chain). The boundary of a k-chain is a $(k-1)$-chain. It is the linear and signed combination of chain element boundary simplices. The space of i-chains is denoted by $C_i(K)$.

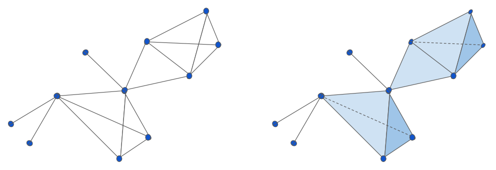


Figure 3: Graph clique complex Cl(G) example.

There are two special cases of chains that will be useful to define homology group:

- Closed chain or i-cycle: i-chain with empty boundary. An i-chain c is an i-cycle if and only if $\partial_i c = 0$, i.e. $c \in ker(\partial_i)$. This subspace of $C_i(K)$ is denoted as $Z_i(K)$.
- Exact chain or i-boundary: An i-chain c is an i-boundary if there exists an $(i+1)$-chain d such that $c = \partial_{i+1}(d)$, i.e. $c \in im(\partial i+1)$. This subspace of $C_i(K)$, the set of all such i-boundaries forms, is denoted by $B_i(K)$.

Now, if we consider i-cycles not bounding an $(i+1)$-simplicial complex, this is the definition of an i-th homology group of the simplicial complex K. The precise definition is the quotient group of $B_i(K)$ module $Z_i(K)$ (i.e. $B_i(K)/Z_i(K)$, see Supplementary Material). The number of non equivalent i-cycles (Figure 5) is the dimension of the homology group $H_i(K)$, also named Betti numbers.

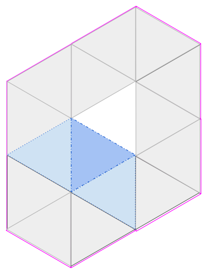


Figure 5: The two blue dashed cycles are homologically equivalent, the pink isn't.

We can create a nested family of simplicial complexes, K_ε, where at each step t, K_{ε_t} is embedded in the simplicial complex $K_{\varepsilon_{t+1}}$. We call this set a simplicial complex filtration.

For each filtration simplicial complex, we can calculate the homology groups. Then, we can look at the birth, that is, when a homology class appears, and death, the time when the homology class disappears. The PH treats the birth and the death of these homological features in K_ε for different ε values. The lifespan of each homological feature can be represented as an interval $(birth, death)$, of the homological class. Given a filtration, this collection of intervals is named a Persistence Diagram (PD) Carlsson (2009).

It is possible to compare two PDs using specific distances (Wasserstein and Bottleneck). To efficiently perform this operation, due to the size of these diagrams, it is sometimes necessary to simplify them by means of a discretization process (such as Weighted Silhouette and Heat vectorizations).

3 Related Work

Algebraic Topology and Machine Learning The use of Algebraic Topology in the fields of data science and machine learning has been gaining momentum in recent years (see Carlsson (2009)). Specifically in the case of neural networks, some works have applied topology for improving the training procedure of the models Hofer et al. (2020); Clough et al. (2020), or pruning the model afterwards Watanabe & Yamana (2020b). Other works have focused on analyzing the capacity of neural networks Guss & Salakhutdinov (2018a); Rieck et al. (2019b); Konuk & Smith (2019) or the complexity of input data Konuk & Smith (2019). Furthermore, recent works have provided topological analysis of the decision boundaries of classifiers based on PH and Betti numbers Ramamurthy et al. (2019); Naitzat et al. (2020).

Graph and topological representations of neural networks Gebhart et al. (2019) suggest a method for computing the PH over the graphical activation neural networks, while Watanabe & Yamana (2020a) propose representing neural networks via simplicial complexes based on Taylor decomposition, from which one can compute the PH. Chowdhury et al. (2019) show that directed homology can be used to represent MLPs. Anonymous (2021) concurrently show neural networks, when represented as directed, acyclic graphs, can be associated to an Algebraic Topology object, specifically to a directed flag complex. By computing the PH diagram, one can effectively characterize neural networks, and even compute distances between two given neural networks, which can be used to measure their similarity. This is unlike other works Corneanu et al. (2019); Guss & Salakhutdinov (2018b) approximating neural networks representations with regard to the input space. Relevant to our work, in Rieck et al. (2019b) authors propose a complexity measure of neural networks based on persistent homology. However, we will see that their representation does not fulfill our requirements in Section 4.

Estimating the generalization and studying the learning process We are, though, specifically interested in the use of PH for analyzing the learning process, especially with the goal of estimating generalization. In this regard, the literature is perhaps more limited. Jiang et al. (2019) work on understanding what drives generalization in deep networks from a Bayesian of view. Neyshabur et al. (2017) study the generalization gap prediction from the training data and network parameters using a margin distribution, which are the distances of training points to the decision boundary. In Li et al. (2020), authors propose an alternative to cross-validation for model selection based on training once on the whole train set, without any data split, deriving a validation set with data augmentation.

Corneanu et al. (2020) try to estimate the performance gap between training and testing using PH measures. They claim. However, one can observe some caveats. The first one is that their regression fitted to predict the test error has a considerably high error, making it not usable in practice. The second caveat is that for fitting the regression one needs at least part of the sequestered testing set.

In this work, motivated by the interest of having a better understanding of whether it would be possible to estimate the generalization of neural networks without a holdout set, we suggest using the topological characterization and distances concurrently proposed in Anonymous (2021) but, crucially, measured between consecutive weight updates. We will show that the evolution of this distance is similar to the one of the validation accuracy. Unlike Li et al. (2020), we do not use any data at all. Unlike Corneanu et al. (2020), we do not build a statistical or machine learning model (linear regression) for predicting the testing error. Instead, we propose a new measure, and we empirically show that it highly correlates with the validation accuracy. Note that in this work we do not work with any input data and activations, but with the parameters of the neural network themselves. The code and outputs are fully available in the Supplementary Material under a MIT License.

4 Approach

Representation For representing neural networks as graphs, we follow the approach proposed concurrently in Anonymous (2021). We associate to the neural network, at each learning state (defined by its weights), a weighted directed graph that is analyzed as an abstract simplicial complex. It is important to note that abstract simplicial complex are used in opposition to geometric simplicial complex (e.g. Vietoris-Rips complex).

For every training state, neural network connections are considered as directed and weighted edges between neurons, represented by graph nodes. Biases are considered as new edges that join to isolate vertices. In this representation, activation functions are lost. Bias information could also have been ignored because, as we will see, it is not very informative in terms of homology, but we decided to preserve it.

Negative edge weights are represented with reverse edges with the same weight absolute value. We discard the use of the absolute value of weights as neural networks are not invariant to weight sign transformations. This representation is consistent with the fact that every neuron can be replaced by a neuron from which two edges with opposite weights emerge and converge again on another neuron with opposite weights. From an homological point of view, this would be represented as a closed cycle. Weights are normalized following the Equation 1. ζ is an smoothing parameter that we set to 1e-6. This smoothing parameter is necessary as we want to avoid normalized weights of edges to be 0 (in our representation 0 implies a lack of connection):

$$max(1 - \frac{|w|}{max(|W|)}, \zeta) \tag{1}$$

Unlike Rieck et al. (2019b) we do not represent the neural network as a multipartite graph with a persistent homology filtration that contains at most 1-simplices (edges), which only capturrd zero-dimensional topological information, i.e. connectivity information. We do not believe the strong assumption that the NNs encode the learned information layer pairwise exclusively since there are trivial global transformations of a NN that are not captured by analyzing pairs of layers, more specifically 1. Superfluous cycle insertions: for instance, add two neurons and connect their input to a single neuron in a lower layer and their two outputs to a single output neuron in an upper layer with opposite weights; 2. Identity layer insertions: insert an intermediate identity layer with neurons and trivially connect to the next layer; and 3. Non-planar neural networks analysis: the analysis of neural networks that use multiple connections between non-consecutive layers require higher order topological analysis.

Algebraic Topology object For each weighted directed graph associated with the state of a neural network, we link a directed flag complex to it. The topological properties of this directed flag complex are studied using homology groups H_n. We calculate the homology groups up to degree 3 (H_0-H_3).

For each state, we use a family of simplicial complexes, K_ε, for a range of values of $\varepsilon \in \mathbb{R}$. The simplicial complex at step ε_t is embedded in the complex at ε_{t+1}, for $\varepsilon_t \leq \varepsilon_{t+1}$, i.e. $K_\varepsilon \subseteq K_{\varepsilon_{t+1}}$. ε is used as a filter that establish the minimum weight of the graph representation edges included on the simplicial complex. This collection of contained simplicial complex (associated to a directed weighted graph), called filtration, $K_{\varepsilon_{min}} \subseteq \ldots \subseteq K_{\varepsilon_t} \subseteq K_{\varepsilon_{t+1}} \subseteq \ldots \subseteq K_{\varepsilon_{max}}$, where $t \in [0,1]$ and $\varepsilon_{min} = 0$, $\varepsilon_{max} = 1$ (remember that edge weights are normalized).

The sequence of homology groups is calculated by varying the ε parameter to obtain the persistence homology diagram. In our case, persistent homology calculations are performed on $\mathbb{Z}_2$. In other words, once the corresponding filter has been applied to the weight of the edges, all connected edges are considered equally.

Distances between persistence diagrams of consecutive states In this paper, we are interested in comparing PDs between different simplicial complex associated to each training state of the neural network. There are two distances traditionally used to compare PDs, Bottleneck distance (the length of the longest edge) and Wasserstein distance (using the sum of all edges lengths, instead of the maximum). Their stability with respect to perturbations on PDs has been object of different studies Chazal et al. (2012); Cohen-Steiner et al. (2005). As shown in comparative studies such as in **?**, different distances and different ways of vectorizing persistence diagrams have results with different levels of stability and quality.

In order to make computations feasible and obviate noisy intervals, we filter the PDs by limiting the minimum PD interval size. We do so by setting a minimum threshold $\eta = 0.01$. Intervals with a lifespan under this value are not considered (spurious homological features). Additionally, for computing distances, we need to remove infinity values. As we are only interested in the deaths until the maximum weight value, we replace all the infinity values by 1.0.

In our case, neural networks have millions of persistence intervals per PD, while Wasserstein distance calculations are computationally hard for large PDs. To make it computationally feasible, we will use a vectorized version of PDs, also called PD discretization. This vectorized version summaries have been proposed and used on recent literature Adams et al. (2017); Berry et al. (2020); Bubenik (2015); Lawson et al. (2019); Rieck et al. (2019a). For persistence diagram distance calculation, we use weighted Silhouette and Heat vectorizations, using the Giotto-TDA library Tauzin et al. (2020).

5 EXPERIMENTS

Data We validate our method in several heterogeneous (vision, natural language), well-known datasets, namely 1. MNIST LeCun & Cortes (2010), 2. CIFAR-10, 3. CIFAR-100 Krizhevsky (2009), and 4. the Reuters dataset Thoma (2017) (multi-class and multi-label document classification dataset).

Models We experiment with two neural architectures,1. MLPs and 2. CNNs. In the latter case, we use the convolutional layers as a pre-trained model with frozen weights, and we learn an MLP on top of it. The reason we do so is that our method is based in a representation that, at least in the basic form, does not allow capturing information from convolutional layers. Thus, we need a single (same weights) feature extractor, to abstract away distances related to the CNN layers and focus on the MLP.

Conducted experiments We define the *base* MLP architecture as `{Input, Linear(512), Dropout(0.2), Linear(512), Dropout(0.2), Output}`. In the case of CNNs, the pre-trained model is defined as 3 convolutional blocks with kernel size 3 (starting with 32 channels), interleaved with max pooling (its linear layers are thrown away after the pre-training). On top of the pre-trained CNN, we also define the same base MLP architecture.

Then, for each dataset and model (MLP and CNN), we experiment with varying (while keeping the rest fixed to the base architecture) 1. Layer size (number of units per layer): 4, 16, 32, 128, 256; 2. Number of labels (the other classes are removed): 2, 4, 6, 8, 10; 3. Learning rate: 1e-e05, 0.0001, 0.001, 0.01, 0.1; 4. Dropout: 0.0, 0.2, 0.4, 0.5, 0.8; and 5. Input order: 5 random reordering of the input samples. As a control experiment, for each analyzed problem we run the same configuration with 5 different input orders. If the measured distances are, indeed, related with the learning process of neural networks, these variations should not have any noticeable effect. We run each configuration 5 times with different random seeds (and, thus, weight initializations[1]) to see if the results are consistent across runs. All models are trained with the RMSProp optimizer with a batch size of 256.

Distances and validation accuracy computation Note that homological distances are obtained at the end of each batch, while validation metrics are only computed on each epoch. We consider the validation curve to be our baseline, and therefore we compare our method to it. The methodology is summarized as follows:

1. In each training step (i.e., batch) we extract the weights from the MLP current state and use them to build an abstract simplicial complex from the associated weighted directed graph.
2. We calculate the homological persistence diagram of the simplicial complex.
3. We then calculate the distance between consecutive persistence diagrams (we will call this sequence *homological convergence*). We use two different distances, Heat and Silhouette.
4. We compare the homological convergence with the evolution of the validation results on neural network learning process.

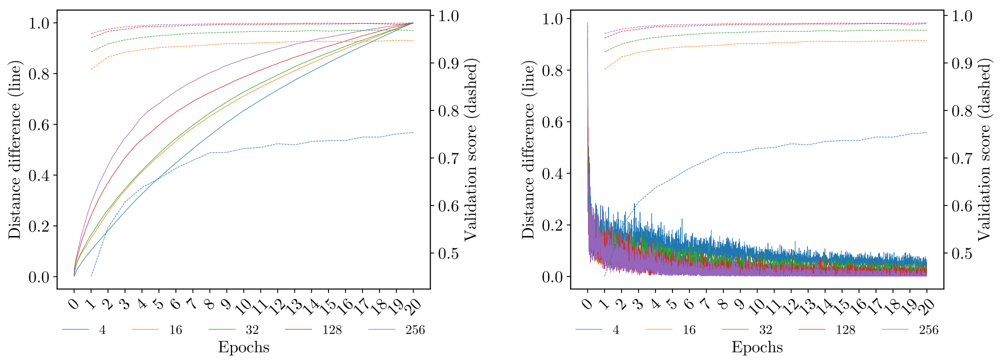
All experiments were executed in a machine with 2 NVIDIA V100 of 32GB, 2 Intel(R) Xeon(R) Platinum 8176 CPU @ 2.10GHz, and of 1.5TB RAM, for a total of around 7 days. We note that our method is considerably demanding in terms of both compute and memory.

6 RESULTS

In this section, we highlight the main results, omitting Silhouette (since the results were clearer with Heat). See the Supplementary Material for all the plots and correlations, including Silhouette.

[1]The pre-trained convolutional weights are always identical, though.

We study the relation between the evolution of the PH diagram distances with the one of the validation score with the cumulative values of the distance between homologous persistence diagrams because this value seems much more stable. The information of the distance between the persistence diagrams has been normalized to visualize clearly the type of evolution of each curve on the same scale. Some of the non-normalized plots can be found in the Supplementary Material. Figure 6 shows the cumulative and non-cumulative homology the MNIST experiment with layer size.



(a) Cumulative

(b) Non-cumulative

Figure 6: Heat distance and validation accuracy curves on MNIST with layer size. Normalized.

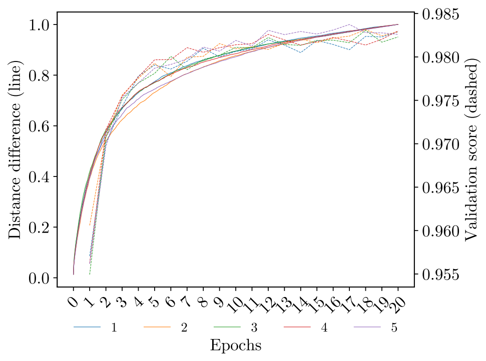
For each experiment (e.g., layer size in MNIST), we plot both the evolution of the PH diagram distance and the validation score (accuracy). The plotted values are the corresponding means of the 5 repetitions with different seeds. In addition, we compute the Pearson correlation for these values. Plots show on the x-axis each training step (for each batch) of the evolution in the training state of the neural network. On the y-axis, two scales are shown that apply to the distance curves between accumulated persistence diagrams (solid lines), scale on the right side, and the neural network validation (dotted lines), numerical scale on the left side. For each sub-experiment (for example, different values of layer size) a different color was used.

The general result is that the evolution of the homological convergence of the MLPs seems to be very similar to the one of the validation score. This is generally consistent across experiments (see the Supplementary Material). Table 1 shows the mean (and standard deviations) of the Pearson correlations for all datasets. All means are above 0.8, implying that there is strong correlation. Intuitively, this is also observed in the plots, although once the distances are normalized it is not as clear to visualize. Interestingly, we find that the very few exceptions in which the correlation is low corresponds to extreme values (very small number of neurons per layer, very high learning rate, very high dropout), in which the neural network doesn't end up learning properly.

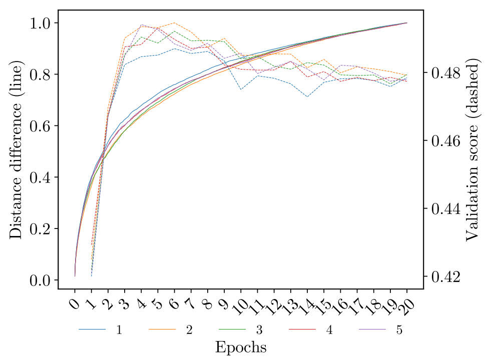
In CNNs, the correlations are lower (but still usually above 0.8 in experiments such as the one of increasing the number of layers). Recall that we froze a convolutional feature extractor since our method only supports MLPs. We believe these lower correlations occur because an important part of the learning process happened in the convolutional layers (in the pre-training), not captured.

Another finding is that the method obtains consistent results across runs, meaning that it is capturing information related to important properties of the networks themselves instead of random artifacts. When varying the studied hyperparameters, we observe that the curves for each configuration are indeed, different. Remarkably, in the control experiments, this is not the case; results show that the homological convergence during the learning of the same problem with the same model but with different input order is very similar. The alteration of the order of the input doesn't have any effect in the homological convergence. The results of two of these experiments are shown in Figure 7.

In addition, we observe that when the neural network learns the given problem, homological convergence occurs. For example, when the layer size is modified, the capacity of the neural network to learn the problem changes (Figure 6). When it can't learn the problem, because the network does not have sufficient capacity (the layer size is too small, 4 units), the homology does not seem to converge.



(a) MNIST



(b) CIFAR-100 CNN

Figure 7: Learning evolution on input order experiments (control experiments). Normalized.

	Heat distance		Silhouette distance	
Dataset	Means mean	Deviations mean	Means mean	Deviations mean
MNIST	0.8910	0.0424	0.8910	0.0424
Reuters	0.6220	0.0700	0.6220	0.0700
CIFAR-10 MLP	0.8233	0.0649	0.8233	0.0649
CIFAR-10 CNN	0.4241	0.1915	0.4241	0.1915
CIFAR-100 MLP	0.8420	0.0566	0.8420	0.0566
CIFAR-100 CNN	0.6130	0.0800	0.6130	0.0800

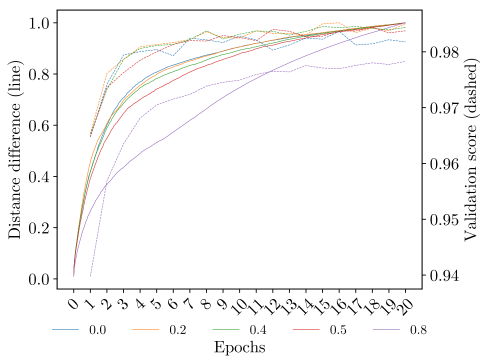
Table 1: Correlation (c. with 20 points) of validation values with topological difference (cumulative).

Regarding the learning rate, the results are coherent with the intuition that it is a fundamental parameter that controls how much to change the model in response to the estimated error during the learning process. A too small learning rate may result in a long training process that could be stalled, while a too large value may fall in a fast suboptimal solution or an unstable training process. Using homological convergence we find similar behaviour, as can be seen in Figure 9.

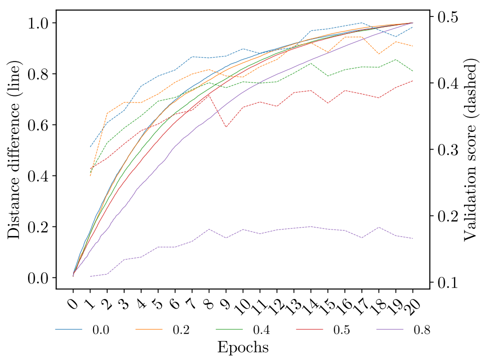
Finally, we note that even if the two convergences (validation and homological convergence) are correlated, they are not the same process. This is especially visible in the learning rate experiments. For instance, in Figure 9, homological convergence is reached before the stabilization of the validation accuracy. Presumably, they are not capturing the exact same information; we believe that the difference is due to the fact that the validation accuracy depends on the specifics of the data sampled in the validation subset, while the homological convergence is independent of the validation data.

7 Discussion

We posed the question whether homological convergence (in terms of distances between PH diagrams in consecutive neural network states) is related to the learning process of neural networks. We have seen that, indeed, it is the case, with strong empirical results backing our claim. This finding has a remarkable implication. If homological convergence evolution mirrors the validation accuracy curve, one could ignore the validation set to monitor the training. This opens the path towards estimating the generalization of neural networks without the need of any holdout set. Researchers have wondered for a long time whether generalization could be predicted from intrinsic properties of the model or training data alone, and in fact other works have claimed to do so. Although we do not provide any predictive model, we show that our proposed measures strongly correlate with validation accuracy. We do so by not using any data at all; we just look at the neural network itself. Our contribution aims pushing towards a better understanding of the learning process of neural networks, not targeting any specific direct application. However, we note that it can be effectively used for monitoring the training of neural networks in terms of convergence expected generalization, as we have extensively

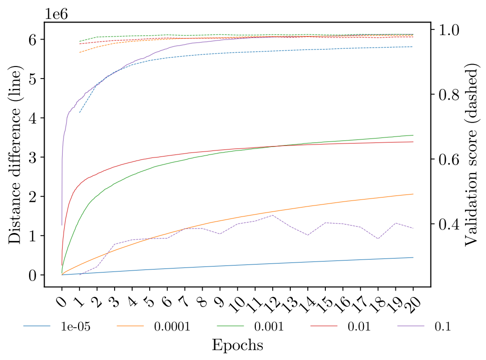


(a) MNIST

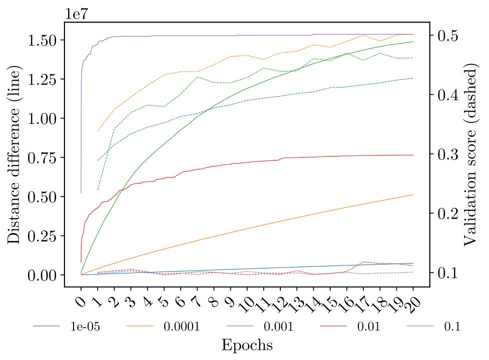


(b) CIFAR-10 MLP

Figure 8: Learning evolution when dropout parameter is changed. Normalized.



(a) MNIST



(b) CIFAR-10 MLP

Figure 9: Learning evolution when modifying the learning rate parameter. Not normalized.

shown in the experiments. Apart from when there is no access to a validation set, this is relevant because depending on a validation set has the risk of overfitting to it. Having an intrinsic measure should be more robust to random noise in a specific data sample.

The main limitation of our method is its computational scalability. This lack of scalability has prevented us from validating our method on bigger models and datasets. However, we note that our approach computes the *exact* PD distances, that is, we do not simplify the graph representation of the neural networks (we keep every single neuron and connections) and we do not approximate any computation. This leaves room for finding efficient approximations, opening a new research line.

Finally, we note that instead of computing correlations, serving as a basic quantitative study, it would be interesting to perform a time-series analysis to gain more insights on how the two curves vary together. Moreover, it would have been interesting to investigate how to build a predictive model of the validation accuracy from the PH distances, but it is was of the scope of this work.

8 Conclusions & Future Work

In this work, we have provided an empirical proof of the fact that homological convergence is related to the learning process and generalization properties of neural networks. Furthermore, we have shown that it can be used to monitor the training of a neural network (and potentially estimating its generalization) without a validation set. As future work, we suggest generalizing our representation to other neural architectures and scaling up the experiments to larger models and datasets, for which finding efficient approximations of our method will be crucial.

REFERENCES

H. Adams, Tegan Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, Sofya Chepushtanova, E. Hanson, F. Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *J. Mach. Learn. Res.*, 18:8:1–8:35, 2017.

Anonymous Anonymous. Characterizing and measuring the similarity of neural networks with persistent homology, 2021.

E. Berry, Yen-Chi Chen, J. Cisewski-Kehe, and Brittany Terese Fasy. Functional summaries of persistence diagrams. *Journal of Applied and Computational Topology*, 4:211–262, 2020.

Peter Bubenik. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16:77–102, 2015.

G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46:255–308, 2009.

F. Chazal, V. D. Silva, and S. Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173:193–214, 2012.

Samir Chowdhury, T. Gebhart, Steve Huntsman, and Matvey Yutin. Path homologies of deep feedforward networks. *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1077–1082, 2019.

J. Clough, I. Öksüz, Nicholas Byrne, V. Zimmer, J. A. Schnabel, and A. P. King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2020.

D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Proceedings of the twenty-first annual symposium on Computational geometry*, 2005.

C. Corneanu, M. Madadi, S. Escalera, and A. Martínez. Computing the testing error without a testing set. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2674–2682, 2020.

C. A. Corneanu, M. Madadi, S. Escalera, and 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)*, pp. 4752–4761, 2019. doi: 10.1109/CVPR.2019.00489.

T. Gebhart, Paul Schrater, and A. Hylton. Characterizing the shape of activation space in deep neural networks. *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1537–1542, 2019.

William H. Guss and R. Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *ArXiv*, abs/1802.04443, 2018a.

William H. Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *CoRR*, abs/1802.04443, 2018b. URL `http://arxiv.org/abs/1802.04443`.

C. Hofer, Florian Graf, M. Niethammer, and R. Kwitt. Topologically densified distributions. *ArXiv*, abs/2002.04805, 2020.

Yiding Jiang, Dilip Krishnan, H. Mobahi, and S. Bengio. Predicting the generalization gap in deep networks with margin distributions. *ArXiv*, abs/1810.00113, 2019.

Emir Konuk and K. Smith. An empirical study of the relation between network architecture and complexity. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 4597–4599, 2019.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

P. Lawson, A. Sholl, J. Brown, Brittany Terese Fasy, and C. Wenk. Persistent homology for the quantitative evaluation of architectural features in prostate cancer histology. *Scientific Reports*, 9, 2019.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Weikai Li, Chuanxing Geng, and Songcan Chen. Leave zero out: Towards a no-cross-validation approach for model selection. *CoRR*, abs/2012.13309, 2020. URL `https://arxiv.org/abs/2012.13309`.

Gregory Naitzat, A. Zhitnikov, and L. Lim. Topology of deep neural networks. *J. Mach. Learn. Res.*, 21:184:1–184:40, 2020.

Behnam Neyshabur, Srinadh Bhojanapalli, D. McAllester, and Nathan Srebro. Exploring generalization in deep learning. In *NIPS*, 2017.

Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5351–5360. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/ramamurthy19a.html`.

Bastian Alexander Rieck, F. Sadlo, and H. Leitte. Topological machine learning with persistence indicator functions. *ArXiv*, abs/1907.13496, 2019a.

Bastian Alexander Rieck, Matteo Togninalli, C. Bock, Michael Moor, Max Horn, Thomas Gumbsch, and K. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. *ArXiv*, abs/1812.09764, 2019b.

Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Anibal Medina-Mardones, Alberto Dassatti, and Kathryn Hess. giotto-tda: A topological data analysis toolkit for machine learning and data exploration, 2020.

Martin Thoma. The reuters dataset, July 2017. URL `https://martin-thoma.com/nlp-reuters`.

S. Watanabe and Hayato Yamana. Topological measurement of deep neural networks using persistent homology. In *ISAIM*, 2020a.

Satoru Watanabe and Hayato Yamana. Deep neural network pruning using persistent homology. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 153–156. IEEE, 2020b.