NEURAL PERSISTENCE: A COMPLEXITY MEASURE FOR DEEP NEURAL NETWORKS USING ALGEBRAIC TOPOLOGY

Anonymous authors
Paper under double-blind review

ABSTRACT

While many approaches to make neural networks more fathomable have been proposed, they are restricted to interrogating the network with input data. Measures for characterizing and monitoring structural properties, however, have not been developed. In this work, we propose *neural persistence*, a complexity measure for neural network architectures based on topological data analysis on weighted stratified graphs. To demonstrate the usefulness of our approach, we show that *neural persistence* agrees with best practices developed in the deep learning community such as dropout and batch normalization. Moreover, we derive a neural persistence-based stopping criterion that shortens the training process while preserving accuracy compared to validation-loss based early stopping.

1 INTRODUCTION

The practical successes of deep learning in various fields such as image processing (Simonyan and Zisserman, 2014; He et al., 2016; Hu et al., 2017), biomedicine (Ching et al., 2018; Rajpurkar et al., 2017; Rajkomar et al., 2018), and language translation (Bahdanau et al., 2014; Sutskever et al., 2014; Wu et al., 2016) still outpace our theoretical understanding. While hyperparameter adjustment strategies exist (Bengio, 2012), formal measures for assessing the generalization capabilities of deep neural networks have yet to be identified (Zhang et al., 2016). Previous approaches for improving theoretical and practical comprehension focus on interrogating networks with input data. These methods include i) feature visualization of deep convolutional neural networks (Zeiler and Fergus, 2014; Springenberg et al., 2014), ii) sensitivity and relevance analysis of features (Montavon et al., 2017), iii) a descriptive analysis of the training process based on information theory (Tishby and Zaslavsky, 2015; Shwartz-Ziv and Tishby, 2017; Saxe et al., 2018; Achille and Soatto, 2017), and iv) a statistical analysis of interactions of the learned weights (Tsang et al., 2017). Additionally, Raghu et al. (2016) develop a measure of *expressivity* of a neural network and use it to explore the empirical success of batch normalization, as well as for the definition of a new regularization method. They note that one key challenge remains, namely to provide meaningful insights while maintaining theoretical generality. This paper presents a method for elucidating neural networks in light of both aspects.

We develop *neural persistence*, a novel measure for characterizing neural network structural complexity. In doing so, we adopt a new perspective that integrates both network weights and connectivity while not relying on interrogating networks through input data. Neural persistence builds on computational techniques from algebraic topology, specifically topological data analysis (TDA), which was already shown to be beneficial for feature extraction in deep learning (Hofer et al., 2017) and describing the complexity of GAN sample spaces (Khrulkov and Oseledets, 2018). More precisely, we rephrase deep networks with fully-connected layers into the language of algebraic topology and develop a measure for assessing the structural complexity of i) individual layers, and ii) the entire network. In this work, we present the following contributions:

- We introduce *neural persistence*, a novel measure for characterizing the structural complexity of neural networks that can be efficiently computed.
- We prove its theoretical properties, such as upper and lower bounds, thereby arriving at a normalization for comparing neural networks of varying sizes.

- We demonstrate the practical utility of neural persistence in two scenarios: i) it correctly captures the benefits of dropout and batch normalization during the training process, and ii) it can be easily used as a competitive early stopping criterion that does not require validation data.

2 BACKGROUND: TOPOLOGICAL DATA ANALYSIS

Topological data analysis (TDA) recently emerged as a field that provides computational tools for analysing complex data within a rigorous mathematical framework that is based on *algebraic topology*. This paper uses persistent homology, a theory that was developed to understand high-dimensional manifolds (Edelsbrunner et al., 2002; Edelsbrunner and Harer, 2010), and has since been successfully employed in characterizing graphs (Sizemore et al., 2017; Rieck et al., 2018), finding relevant features in unstructured data (Lum et al., 2013), and analysing image manifolds (Carlsson et al., 2008). Due to space constraints, this section gives a brief summary of the key concepts. Please refer to Edelsbrunner and Harer (2010) for an extensive introduction.

Simplicial homology The central object in algebraic topology is a simplicial complex K, i.e. a high-dimensional generalization of a graph, which is typically used to describe complex objects such as manifolds. Various notions to describe the connectivity of K exist, one of them being simplicial homology. Briefly put, simplicial homology uses matrix reduction algorithms (Munkres, 1996) to derive a set of groups, the homology groups, for a given simplicial complex K. Homology groups describe topological features—colloquially also referred to as holes—of a certain dimension d, such as connected components ($d = 0$), tunnels ($d = 1$), and voids ($d = 2$). The information from the dth homology group is summarized in a simple complexity measure, the dth Betti number β_d, which merely counts the number of d-dimensional features. For example, a circle has Betti numbers $(1, 1)$, i.e. one connected component and one tunnel, while a filled circle has Betti numbers $(1, 0)$, i.e. one connected component but no tunnel. In the context of analysing simple feedforward neural networks for two classes, Bianchini and Scarselli (2014) calculated bounds of Betti numbers of the decision region belonging to the positive class, and were thus able to show the implications of different activation functions.

Persistent homology For the analysis of real-world data sets, however, Betti numbers turn out to be of limited use because their representation is too coarse and unstable. This prompted the development of persistent homology. Given a simplicial complex K with an additional set of weights $a_0 \leq a_1 \leq \cdots \leq a_{m-1} \leq a_m$, which are commonly thought to represent the idea of a scale, it is possible to put K in a filtration, i.e. a nested sequence of simplicial complexes $\emptyset = \mathrm{K}_0 \subseteq \mathrm{K}_1 \subseteq \cdots \subseteq \mathrm{K}_{m-1} \subseteq \mathrm{K}_m = \mathrm{K}$. This filtration is thought to represent the "growth" of K as the scale is being changed. During this growth process, topological features may be *created* (new vertices may be added, for example, which creates a new connected component) or *destroyed* (two connected components may merge into one). Persistent homology tracks these changes and represents the creation and destruction of a feature as a point $(a_i, a_j) \in \mathbb{R}^2$ for indices $i \leq j$ with respect to the filtration. The collection of all points corresponding to d-dimensional topological features is called the dth *persistence diagram* $\mathcal{D}_d$. It can be seen as a collection of Betti numbers at multiple scales. Given a point $(x, y) \in \mathcal{D}_d$, the quantity $\mathrm{pers}(x, y) := |y - x|$ is referred to as its *persistence*. Typically, high persistence is considered to correspond to *features*, while low persistence is seen to indicate *noise* (Edelsbrunner et al., 2002).

3 A NOVEL MEASURE FOR NEURAL NETWORK COMPLEXITY

This section details *neural persistence*, our novel measure for assessing the structural complexity of neural networks. By exploiting both network structure and weight information through persistent homology, our measure captures network expressiveness, going beyond mere connectivity properties. Subsequently, we describe its calculation, provide theorems for theoretical and empirical bounds, and show the existence of neural networks complexity regimes. To summarize this section, Figure 1 illustrates how our method treats a neural network.

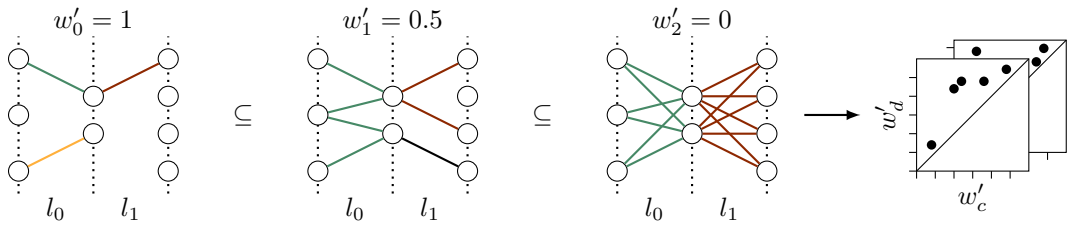

Figure 1: Illustrating the neural persistence calculation of a network with two layers (l_0 and l_1). Colours indicate connected components per layer. The filtration process is depicted by colouring connected components that are created or merged when the respective weights are greater than or equal to the threshold w'_i. As w'_i decreases, the network connectivity increases. Creation and destruction thresholds are collected in one persistence diagram per layer (right), and summarized according to Equation 1 for calculating neural persistence.

3.1 Neural persistence

Given a feedforward neural network with an arrangement of neurons and their connections E, let $\mathcal{W}$ refer to the set of weights. $\mathcal{W}$ is typically changing during training. Therefore, we require a function $\varphi\colon E \to \mathcal{W}$ that maps a specific edge to a weight. Fixing an activation function, the connections form a *stratified graph*.

Definition 1 (Stratified graph and layers)**.** *A* stratified graph *is a multipartite graph $G = (V, E)$ satisfying $V = V_0 \sqcup V_1 \sqcup \ldots$, such that if $u \in V_i$, $v \in V_j$, and $(u, v) \in E$, we have $j = i + 1$. Hence, edges are only permitted between adjacent vertex sets. Given $k \in \mathbb{N}$, the kth* layer *of a stratified graph is the unique subgraph $G_k := (V_k \sqcup V_{k+1}, E_k := E \cap \{V_k \times V_{k+1}\})$.*

This enables calculating the persistent homology of G and each G_k, using the filtration induced by sorting all weights, which is common practice in topology-based network analysis (Carstens and Horadam, 2013; Horak et al., 2009) where weights often represent closeness or node similarity. However, our context requires a novel filtration because the weights arise from an incremental fitting procedure, namely the training, which could theoretically lead to unbounded values. When analysing geometrical data with persistent homology, one typically selects a filtration based on the (Euclidean) distance between data points (Bubenik, 2015). The filtration then connects points that are increasingly distant from each other, starting from points that are direct neighbours. Our network filtration aims to mimic this behaviour in the context of fully-connected neural networks. Our framework does not *explicitly* take activation functions into account; however, activation functions influence the evolution of weights during training.

Filtration Given the set of weights $\mathcal{W}$ for one training step, set $w_{\max} := \max_{w \in \mathcal{W}} |w|$. Furthermore, let $\mathcal{W}' := \{|w|/w_{\max} \mid w \in \mathcal{W}\}$ be the set of transformed weights, indexed in non-ascending order, such that $1 = w'_0 \geq w'_1 \geq \cdots \geq 0$. This permits us to define a filtration for the kth layer G_k as $G_k^{(0)} \subseteq G_k^{(1)} \subseteq \ldots$, where $G_k^{(i)} := (V_k \sqcup V_{k+1}, \{(u, v) \mid (u, v) \in E_k \wedge \varphi'(u, v) \geq w'_i\})$ and $\varphi'(u, v) \in \mathcal{W}'$ denotes the transformed weight of an edge. We tailored this filtration towards the analysis of neural networks, for which large (absolute) weights indicate that certain neurons exert a larger influence over the final activation of a layer. The strength of a connection is thus preserved by the filtration, and weaker weights, i.e. $|w| \approx 0$, remain close to 0. Moreover, since $w' \in [0, 1]$ holds for the transformed weights, this filtration makes the network invariant to scaling, which simplifies the comparison of different networks.

Persistence diagrams Having set up the filtration, we can calculate persistent homology for every layer G_k. As the filtration contains at most 1-simplices (edges), we capture zero-dimensional topological information, i.e. how connected components are created and merged during the filtration. These information are structurally equivalent to calculating a maximum spanning tree using the weights, or performing hierarchical clustering with a specific setup (Carlsson and Mémoli, 2010). While it would theoretically be possible to include higher-dimensional information about each layer G_k, for example in the form of cliques (Rieck et al., 2018), we focus on zero-dimensional information in this paper, because of the following advantages: i) the resulting values are easily

Algorithm 1 Neural persistence calculation

Require: Neural network with l layers and weights $\mathcal{W}$

1: $w_{\max} \leftarrow \max_{w \in \mathcal{W}} |w|$ ▷ Determine largest absolute weight
2: $\mathcal{W}' \leftarrow \{|w|/w_{\max} \mid w \in \mathcal{W}\}$ ▷ Transform weights for filtration
3: **for** $k \in \{0, \ldots, l-1\}$ **do**
4: $\quad F_k \leftarrow G_k^{(0)} \subseteq G_k^{(1)} \subseteq \ldots$ ▷ Establish filtration of kth layer
5: $\quad \mathcal{D}_k \leftarrow \text{PERSISTENTHOMOLOGY}(F_k)$ ▷ Calculate persistence diagram
6: **end for**
7: **return** $\{\|\mathcal{D}_0\|_p, \ldots, \|\mathcal{D}_{l-1}\|_p\}$ ▷ Calculate neural persistence for each layer

interpretable as they essentially describe the clustering of the network at multiple weight thresholds, ii) previous research (Rieck and Leitte, 2016; Hofer et al., 2017) indicates that zero-dimensional topological information is already capturing a large amount of information, and iii) persistent homology calculations are highly efficient in this regime (see below). We thus calculate zero-dimensional persistent homology with this filtration. The resulting persistence diagrams have a special structure: since our filtration solely sorts *edges*, all vertices are present at the beginning of the filtration, i.e. they are already part of $G_k^{(0)}$ for each k. As a consequence, they are assigned a weight of 1, resulting in $|V_k \times V_{k+1}|$ connected components. Hence, entries in the corresponding persistence diagram $\mathcal{D}_k$ are of the form $(1, x)$, with $x \in \mathcal{W}'$, and will be situated *below* the diagonal, similar to superlevel set filtrations (Bubenik, 2015; Cohen-Steiner et al., 2009). Using the p-norm of a persistence diagram, as introduced by Cohen-Steiner et al. (2010), we obtain the following definition for neural persistence.

Definition 2 (Neural persistence)**.** *The neural persistence of the kth layer G_k, denoted by $\mathrm{NP}(G_k)$, is the p-norm of the persistence diagram $\mathcal{D}_k$ resulting from our previously-introduced filtration, i.e.*

$$\mathrm{NP}(G_k) := \|\mathcal{D}_k\|_p := \Big(\sum_{(c,d) \in \mathcal{D}_k} \mathrm{pers}(c,d)^p \Big)^{\frac{1}{p}}, \tag{1}$$

which (for $p = 2$) captures the Euclidean distance of points in $\mathcal{D}_k$ to the diagonal.

The p-norm is known to be a stable summary (Cohen-Steiner et al., 2010) of topological features in a persistence diagram. For it to be a meaningful measure of structural complexity, neural persistence should increase as a neural network is learning and when comparing a given network with a larger architecture. We evaluate these requirements in Section 4.

Algorithm 1 gives a pseudocode description of the calculation process. It is highly efficient: the filtration (line 4) amounts to sorting all n weights of a network, which has a computational complexity of $\mathcal{O}(n \log n)$. Calculating persistent homology of this filtration (line 5) can be realized using an algorithm based on union–find data structures Edelsbrunner et al. (2002). This has a computational complexity of $\mathcal{O}(n \cdot \alpha(n))$, where $\alpha(\cdot)$ refers to the extremely slow-growing inverse of the Ackermann function (Cormen et al., 2009, Chapter 22). We make our implementation and experiments available under `https://osf.io/6zpxa/?view_only=3617b7c6d1cf4699ac7581acb4f4f8f9`.[1]

3.2 Properties of neural persistence

We want to elucidate properties about neural persistence in order to facilitate the comparison of networks with different architectures. As a first step, we derive *bounds* for the neural persistence of a single layer G_k.

Theorem 1. *Let G_k be a layer of a neural network according to Definition 1. Furthermore, let $\varphi_k \colon E_k \to \mathcal{W}'$ denote the unique function that assigns each edge of G_k a transformed weight. Using the filtration from Section 3.1 to calculate persistent homology, the neural persistence $\mathrm{NP}(G_k)$ of the kth layer satisfies*

$$0 \leq \mathrm{NP}(G_k) \leq \left(\max_{e \in E_k} \varphi_k(e) - \min_{e \in E_k} \varphi_k(e) \right) (|V_k \times V_{k+1}| - 1)^{\frac{1}{p}}, \tag{2}$$

[1] The file `Neural_Persistence_ICLR_2019.tar.gz` contains code for reproducing a subset of our experiments.

where $|V_k \times V_{k+1}|$ denotes the cardinality of the vertex set, i.e. the number of neurons in the layer.

Proof. We prove this constructively and show that the bounds can be realized. For the lower bound, let G_k^- be a fully-connected layer with $|V_k|$ vertices and, given $\theta \in [0,1]$, let $\varphi_k(e) := \theta$ for every edge e. Since a vertex v is created before its incident edges, the filtration degenerates to a lexicographical ordering of vertices and edges, and all points in $\mathcal{D}_k$ will be of the form (θ, θ). Thus, $\mathrm{NP}(G_k^-) = 0$. For the upper bound, let G_k^+ again be a fully-connected layer with $|V_k| \geq 3$ vertices and let $a, b \in [0,1]$ with $a < b$. Select one edge e' at random and define a weight function as $\varphi(e') := b$ and $\varphi(e) := a$ otherwise. In the filtration, the addition of the first edge will create a pair of the form (b, b), while all other pairs will be of the form (b, a). Consequently, we have

$$\mathrm{NP}(G_k^+) = \Big(\mathrm{pers}(b,b)^p + (n-1) \cdot \mathrm{pers}(b,a)^p \Big)^{\frac{1}{p}} = (b-a) \cdot (n-1)^{\frac{1}{p}} \tag{3}$$

$$= \left(\max_{e \in E_k} \varphi(e) - \min_{e \in E_k} \varphi(e) \right) (|V_k| - 1)^{\frac{1}{p}}, \tag{4}$$

our upper bound can be realized. To show that this term cannot be exceeded by $\mathrm{NP}(G)$ for any G, suppose we perturb the weight function $\widetilde{\varphi}(e) := \varphi(e) + \epsilon \in [0,1]$. This cannot increase NP, however, because each difference $b - a$ in Equation 3 is maximized by $\max \varphi(e) - \min \varphi(e)$. □

We can use the upper bound of Theorem 1 to normalize the neural persistence of a layer, making it possible to compare layers (and neural networks) that feature different architectures, i.e. a different number of neurons.

Definition 3 (Normalized neural persistence)**.** *For a layer G_k following Definition 1, using the upper bound of Theorem 1, the* normalized *neural persistence $\widetilde{\mathrm{NP}}(G_k)$ is defined as the neural persistence of G_k divided by its upper bound, i.e. $\widetilde{\mathrm{NP}}(G_k) := \mathrm{NP}(G_k) \cdot \mathrm{NP}(G_k^+)^{-1}$.*

The normalized neural persistence of a layer permits us to extend the definition to an entire network. While this is more complex than using a single filtration for a neural network, this permits us to side-step the problem of different layers having different scales.

Definition 4 (Mean normalized neural persistence)**.** *Considering a network as a stratified graph G according to Definition 1, we sum the neural persistence values per layer to obtain the* mean *normalized neural persistence, i.e. $\overline{\mathrm{NP}}(G) := 1/l \cdot \sum_{k=0}^{l-1} \widetilde{\mathrm{NP}}(G_k)$.*

While Theorem 1 gives a lower and upper bound in a general setting, it is possible to obtain empirical bounds when we consider the tuples that result from the computation of a persistence diagram. Recall that our filtration ensures that the persistence diagram of a layer contains tuples of the form $(1, w_i)$, with $w_i \in [0,1]$ being a transformed weight. Exploiting this structure permits us to obtain bounds that could be used prior to calculating the actual neural persistence value in order to make the implementation more efficient.

Theorem 2. *Let G_k be a layer of a neural network as in Theorem 1 with n vertices and m edges whose edge weights are sorted in non-descending order, i.e. $w_0 \leq w_2 \leq \cdots \leq w_{m-1}$. Then $\mathrm{NP}(G_k)$ can be empirically bounded by*

$$\|\mathbb{1} - \mathbf{w}_{\max}\|_p \leq \mathrm{NP}(G_k) \leq \|\mathbb{1} - \mathbf{w}_{\min}\|_p \,, \tag{5}$$

where $\mathbf{w}_{\max} = (w_{m-1}, w_{m-2}, \ldots, w_{m-n})^T$ and $\mathbf{w}_{\min} = (w_0, w_2, \ldots, w_{n-1})^T$ are the vectors containing the n largest and n smallest weights, respectively.

Proof. See Section A.2 in the appendix.

Complexity regimes in neural persistence As an application of the two theorems, we briefly take a look at how neural persistence changes for different classes of simple neural networks. To this end, we train a perceptron on the MNIST data set. Since our measure uses the weight matrix of a perceptron, we can compare its neural persistence with the neural persistence of random weight matrices, drawn from different distributions. Moreover, we can compare trained networks with respect to their initial parameters. Figure 2 depicts the neural persistence values as well as the lower bounds

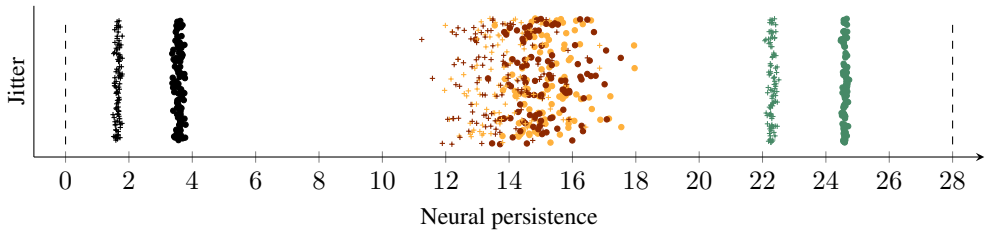

Figure 2: Neural persistence values of trained perceptrons (green), diverging ones (yellow), random Gaussian matrices (red), and random uniform matrices (black). We performed 100 runs per category; dots indicate neural persistence while crosses indicate the predicted lower bound according to Theorem 2. The bounds according to Theorem 1 are shown as dashed lines.

according to Theorem 2 for different settings. We can see that a network in which the optimizer diverges (due to improperly selected parameters) is similar to a random Gaussian matrix. Trained networks, on the other hand, are clearly distinguished from all other networks. Uniform matrices have a significantly lower neural persistence than Gaussian ones. This is in line with the intuition that the latter type of networks induces functional sparsity because few neurons have large absolute weights. For clarity, we refrain from showing the empirical upper bounds because most weight distributions are highly right-tailed; the bound will not be as tight as the lower bound. These results are in line with a previous analysis (Sizemore et al., 2017) of small weighted networks, in which persistent homology is seen to outperform traditional graph-theoretical complexity measures such as the clustering coefficient (see also Figure A.1 in the appendix).

4 Experiments

This section demonstrates the utility and relevance of neural persistence for fully connected deep neural networks. We examine how commonly used regularization techniques (batch normalization and dropout) affect neural persistence of trained networks. Furthermore, we develop an early stopping criterion based on neural persistence and we compare it to the traditional criterion based on validation loss. We used different architectures with *ReLU* activation functions across experiments. The brackets denote the number of units per *hidden* layer. In addition, the Adam optimizer with hyperparameters tuned via cross-validation was used unless noted otherwise. Please refer to Section A.3–A.6 in the appendix for further details about the experiments (such as parameter choices and extended figures).

4.1 Deep learning best practices in light of neural persistence

We compare the mean normalized neural persistence (Definition 4) of a two-layer (with an architecture of $[650, 650]$) neural network to two models where batch normalization (Ioffe and Szegedy, 2015) or dropout (Srivastava et al., 2014) are applied. Figure 3 shows that the networks designed according to best practices yield higher normalized neural persistence values on the MNIST data set in comparison to an unmodified network. The effect of dropout on the mean normalized neural persistence is more pronounced and this trend is directly analogous to the observed accuracy on the test set. These results are consistent with expectations if we consider dropout to be similar to ensemble learning (Hara et al.,

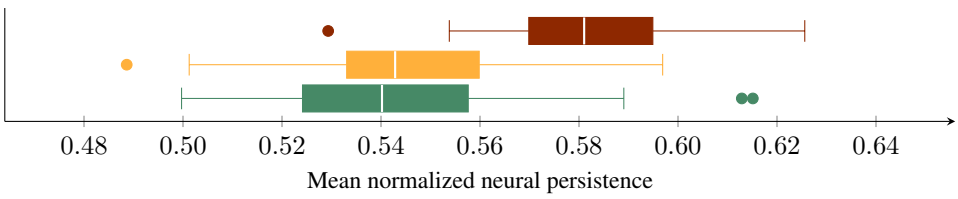

Figure 3: Comparison of mean normalized neural persistence for trained networks without modifications (green), with batch normalization (yellow), and with 50% of the neurons dropped out during training (red) for the MNIST data set (50 runs per setting).

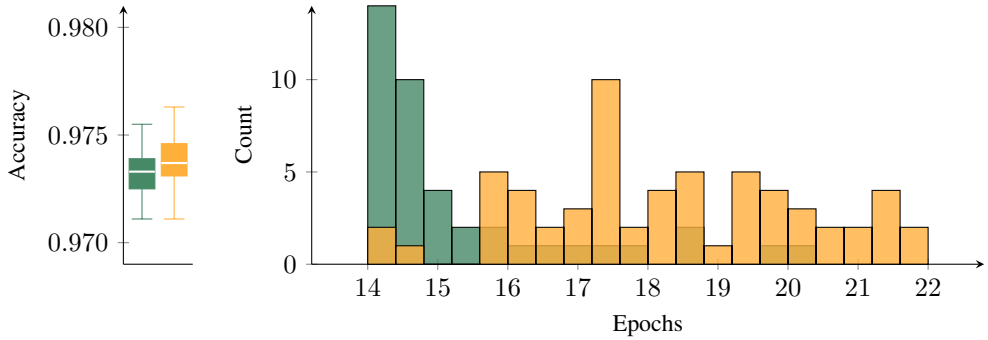

Figure 4: Accuracies (left) obtained with the stopping criterion based on mean normalized neural persistence (green) vs. the one based on validation loss (yellow) for 100 runs on the MNIST data set. The histogram of the number of epochs (right) shows that our criterion tends to stop slightly earlier.

2016). As individual parts of the network are trained independently, a higher degree of per-layer redundancy is expected, resulting in a different structural complexity. Overall, these results indicate that approaches targeted at increasing the neural persistence for a fixed architecture may be of particular interest.

4.2 Early stopping based on neural persistence

Mean normalized neural persistence can be used as an *early stopping* criterion to prevent overfitting: if the mean normalized neural persistence does not increase by more than $\Delta_{\min}$ during a certain number of epochs g, the training process is stopped. This procedure is called "patience"; Algorithm 2 describes the criterion in detail. A similar variant of this algorithm, using validation loss instead of persistence, is the state-of-the-art for early stopping in training (Bengio, 2012; Chollet et al., 2015).

We train a network with a $[50, 50, 20]$ architecture on MNIST, monitored by two early stopping criteria (i.e. one based on validation loss, the other based on mean normalized neural persistence), each of which can stop training. For both criteria, we use $g = 4$ and $\Delta_{\min} = 0$ in order to remain comparable and scale-invariant, as non-zero values of $\Delta_{\min}$ could implicitly favour one of them due to scaling. Figure 4 depicts the resulting accuracies (left) on the test data set, as well as histograms (right) of the epoch at which one of the criteria was triggered first. Out of 100 repetitions, our measure was triggered first 42 times (while it is possible to use different parameters for g and $\Delta_{\min}$ to trigger the criterion based on validation loss first, this demonstrates that our measure results in a *valid* early stopping criterion). We observe that our measure stops slightly earlier while having almost identical test accuracies (the difference in mean accuracy is < 0.0005). In contrast to the classical early stopping criterion, our measure does *not* depend on validation data.

Algorithm 2 Early stopping based on mean normalized neural persistence

Require: Weighted neural network $\mathcal{N}$, patience g, $\Delta_{\min}$

```
P ← 0, G ← 0                                   ▷ Initialize highest observed value and patience counter
procedure EarlyStopping(N, g, Δ_min)           ▷ Callback that monitors training at every epoch
    P' ← NP(N)
    if P' > P + Δ_min then                     ▷ Update mean normalized neural persistence and reset counter
        P ← P', G ← 0
    else                                       ▷ Update patience counter
        G ← G + 1
    end if
    if G ≥ g then                              ▷ Patience criterion has been triggered
        return P                               ▷ Stop training and return highest observed value
    end if
end procedure
```

On a larger architecture, i.e. $[300, 100]$, with the same patience parameters, we stop earlier 63 times out of 100 runs, with an absolute difference in mean accuracy of < 0.0005 (not shown). For the CIFAR-10 data set with an architecture of $[800, 300, 800]$, we observe that both criteria are only functional for $g < 3$ (none of them was triggered for higher values of g). Here, the criterion based on neural persistence results in higher accuracies on average (0.53) than the loss-based one (0.47); please refer to Figure A.3 in the appendix. In light of Theorem 2, we also experimented with using the p-norm of *all* weights of the neural network as a proxy for neural persistence. This did not result in a valid early stopping measure, though, as it was never triggered. This suggests that neural persistence, based on our filtration, captures salient information that would otherwise be hidden among all the weights of a network.

Finally, Section A.6 shows an extended analysis where we subsampled the MNIST data set to simulate a smaller data set. In this case, it is beneficial to free validation data to generate more training samples. We demonstrate that the proposed stopping criterion based on neural persistence gives a slight edge in accuracy over competing methods in the regime of small training data sets because it is capable of using more data for training.

5 Discussion

In this work, we presented *neural persistence*, a novel topological measure of the structural complexity of deep neural networks. We showed that this measure captures topological information that pertains to deep learning performance. Being rooted in a rich body of research, our measure is theoretically well-defined and, in contrast to previous work, generally applicable as well as computationally efficient. We showed that our measure correctly identifies networks that employ best practices such as dropout and batch normalization. Moreover, we developed an early stopping criterion that exhibits competitive performance while not relying on a separate validation data set. Thus, by saving valuable data for training, we managed to boost accuracy, which can be crucial for enabling deep learning in regimes of smaller sample sizes. We plan to extend the measure to more involved architectures, such as CNNs (with pooling layers) and RNNs. Such extensions require additional modifications to our current setup, though. Furthermore, we conjecture that assessing dissimilarities of networks by means of persistence diagrams (making use of higher-dimensional topological features), for example, will lead to further insights regarding their generalization and learning abilities. Another interesting avenue for future research would concern the analysis of the 'function space' learned by a neural network. On a more general level, *neural persistence* demonstrates the great potential of topological data analysis in machine learning.

References

A. Achille and S. Soatto. On the emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 2017.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478. Springer, Heidelberg, 2012.

M. Bianchini and 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):1553–1565, 2014.

P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, Jan. 2015.

G. Carlsson and F. Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11:1425–1470, Apr. 2010.

G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008.

C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013:815035:1–815035:7, 2013.

T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *bioRxiv*, page 142760, 2018.

F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, Feb. 2009. doi: 10.1007/s10208-008-9027-z.

D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko. Lipschitz functions have L_p-stable persistence. *Foundations of Computational Mathematics*, 10(2):127–139, 2010.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 3 edition, 2009.

H. Edelsbrunner and J. Harer. *Computational topology: An introduction*. American Mathematical Society, Providence, RI, USA, 2010.

H. Edelsbrunner, D. Letscher, and A. J. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

K. Hara, D. Saitoh, and H. Shouno. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pages 72–79. Springer, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1633–1643, 2017.

D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034:1–P03034:24, 2009.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

V. Khrulkov and I. Oseledets. Geometry score: A method for comparing generative adversarial networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2621–2629. Proceedings of Machine Learning Research, July 2018.

P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3:1–8, 2013.

G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.

J. R. Munkres. *Elements of algebraic topology*. CRC Press, Boca Raton, FL, USA, 1996.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1 (1):18, 2018.

P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest X-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

B. Rieck and H. Leitte. Exploring and comparing clusterings of multivariate data sets using persistent homology. *Computer Graphics Forum*, 35(3):81–90, 2016.

B. Rieck, U. Fugacci, J. Lukasczyk, and H. Leitte. Clique community persistence: A topological visual analysis approach for complex networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):822–831, 2018.

A. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. Tracey, and D. Cox. On the information bottleneck theory of deep learning. In *Proc. Int. Conf. on Learning Representations (ICLR). Vancouver (May 2018), openreview. net/forum*, 2018.

R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

A. Sizemore, C. Giusti, and D. S. Bassett. Classification of weighted networks through mesoscale homological features. *Journal of Complex Networks*, 5(2):245–273, 2017.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

M. Tsang, D. Cheng, and Y. Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on Computer Vision*, pages 818–833, Heidelberg, Germany, 2014. Springer.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A APPENDIX

A.1 COMPARISON WITH GRAPH-THEORETICAL MEASURES

Traditional complexity/structural measures from graph theory, such as the clustering coefficient, the average shortest path length, and global/local efficiency are already known to be insufficiently accurate to characterize different models of complex random networks Sizemore et al. (2017). Our experiments indicate that this holds true for (deep) neural networks, too. As a brief example, we trained a perceptron on the MNIST data set with batch stochastic gradient descent ($\eta = 0.5$), achieving a test accuracy of ≈ 0.91. Moreover, we intentionally "sabotaged" the training by setting $\eta = 1 \times 10^{-5}$ such that SGD is unable to converge properly. This leads to networks with accuracies ranging from 0.38–0.65. A complexity measure should be capable of distinguishing both classes of networks. However, as Figure A.1 (top) shows, this is *not* the case for the clustering coefficient. Neural persistence (bottom), on the other hand, results in two regimes that can clearly be distinguished, with the trained networks having a significantly smaller variance.

A.2 PROOF OF THEOREM 2

Proof. We may consider the filtration from Section 3.1 to be a subset selection problem with constraints, where we select n out of m weights. The neural persistence $\mathrm{NP}(G_k)$ of a layer thus only depends on the *selected* weights that appear as tuples of the form $(1, w_i)$ in $\mathcal{D}_k$. Letting $\widetilde{\mathbf{w}}$ denote the vector of selected weights arising from the persistence diagram calculation, we can rewrite neural persistence as $\mathrm{NP}(G_k) = \|\mathbb{1} - \widetilde{\mathbf{w}}\|_p$. Furthermore, $\widetilde{\mathbf{w}}$ satisfies $\|\mathbf{w}_{\min}\|_p \leq \|\widetilde{\mathbf{w}}\|_p \leq \|\mathbf{w}_{\max}\|_p$. Since all transformed weights are non-negative in our filtration, it follows that (note the reversal of the two terms)

$$\|\mathbb{1} - \mathbf{w}_{\max}\|_p \leq \mathrm{NP}(G_k) \leq \|\mathbb{1} - \mathbf{w}_{\min}\|_p \,, \tag{6}$$

and the claim follows. □

A.3 EXPERIMENTAL SETUP

Here we give details about our experimental setup. Our aim is to elucidate the properties of our measure by means of suitably realistic architectures. Architectures, optimizers, and their respective hyperparameters are shown for our experiments on deep learning best practices (Table A.1) and early stopping (Table A.2).

A.4 TESTING ACCURACY OF DIFFERENTLY REGULARIZED MODELS

We showed in the main text that neural persistence is capable of distinguishing between networks trained with/without batch normalization and/or dropout. Figure A.2 additionally shows test set accuracies.

A.5 EARLY STOPPING

As stated in the paper, early stopping based on our criterion results in improved accuracies for the CIFAR-10 data set. We reported the accuracies in the main text. However, a boxplot (Figure A.3) shows the significant difference between the means.

A.6 EARLY STOPPING WITH A VALIDATION DATA SET

Labelled data is scarce in most domains of interest. If the validation dataset is used to prevent overfitting, a measure relying purely on internal information and not on the validation loss frees data for the training process. We compare the accuracy and the training duration of a deep network ($[50, 50, 20]$ architecture) on the test set of MNIST while varying the size of the training and validation set. The following measures are used for early stopping with $g = 4$ (modification in Line 3 of Algorithm 2): i) Validation loss ii) Training loss iii) Neural persistence iv) Early stopping at $g = 4$. For all measures except validation loss, the validation dataset (20%) is included in the training process.

Figure A.4 shows the results averaged over 20 runs (the error is the standard deviation). The x-axis depicts the fraction of MNIST provided to the algorithm. The left-hand side plot reports the test accuracy of the final network while the right-hand side plot shows the number of epochs the network is trained. From the results it is clear that more data yields higher accuracy. Moreover, stopping directly after $g = 4$ epochs results in inferior accuracy. Lastly, using the proposed persistence based stopping criterion for early stopping in training gives a slight edge in accuracy over the competitor methods in the regime of few labeled samples.

Table A.1: Parameters and hyperparameters for the experiment on best practices and neural persistence. Dropout and batch normalization were applied after the first hidden layer. Throughout the networks, *ReLU* was the activation function of choice.

Data set	# Runs	# Epochs	Architecture	Optimizer	Batch Size	Hyperparameters
MNIST	50	40	$[650, 650]$	Adam	32	$\eta = 0.0003$ $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$
						$\eta = 0.0003$ $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, Batch Normalization
						$\eta = 0.0003$ $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, Dropout 50%

Table A.2: Parameters and hyperparameters for the experiment on early stopping. Throughout the networks, *ReLU* was the activation function of choice.

Data set	# Runs	# Epochs	Architecture	Optimizer	Batch Size	Hyperparameters
MNIST	100	10	Perceptron	Minibatch SGD	100	$\eta = 0.5$
		40	$[50, 50, 20]$	Adam	32	$\eta = 0.0003$ $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$
			$[300, 100]$			
CIFAR-10	10	80	$[800, 300, 800]$	Adam	128	$\eta = 0.0003$ $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$

A.7 FIGURES

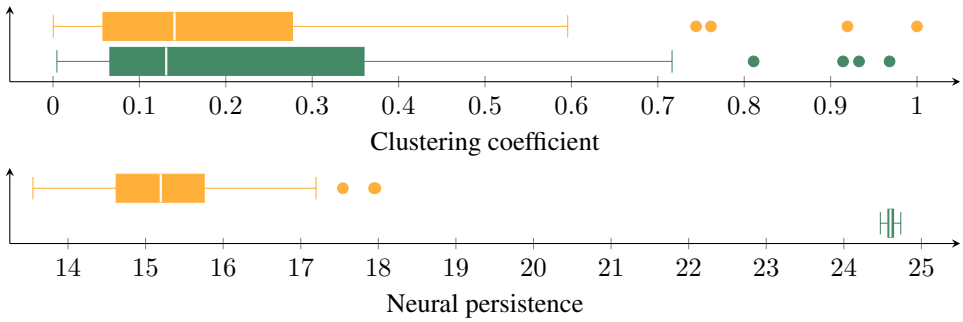

Figure A.1: Traditional graph measures (top), such as the clustering coefficient, fail to detect differences in the complexity of neural networks. Our novel *neural persistence* measure (bottom), by contrast, shows that trained networks with $\eta = 0.5$ (green), which have an accuracy of ≈ 0.91, obey a different distribution than networks trained with $\eta = 1 \times 10^{-0.5}$ (yellow), which have accuracies ranging from 0.38–0.65.

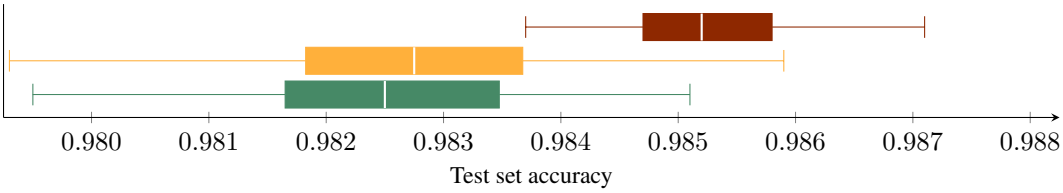

Figure A.2: Comparison of test set accuracy for trained networks without modifications (green), with batch normalization (yellow), and with 50% of the neurons dropped out during training (red) for the MNIST data set.

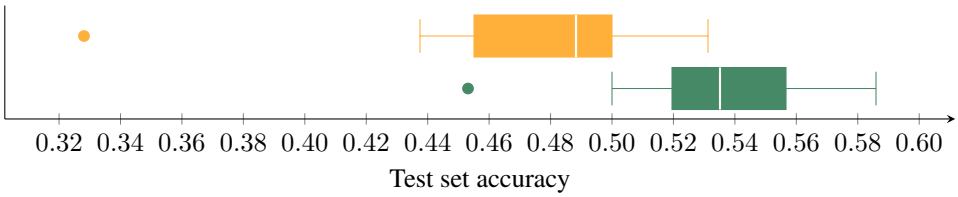

Figure A.3: Test set accuracy achieved when doing early stopping based on mean normalized neural persistence (green) vs. the criterion based on validation loss (yellow) on the CIFAR-10 data set.

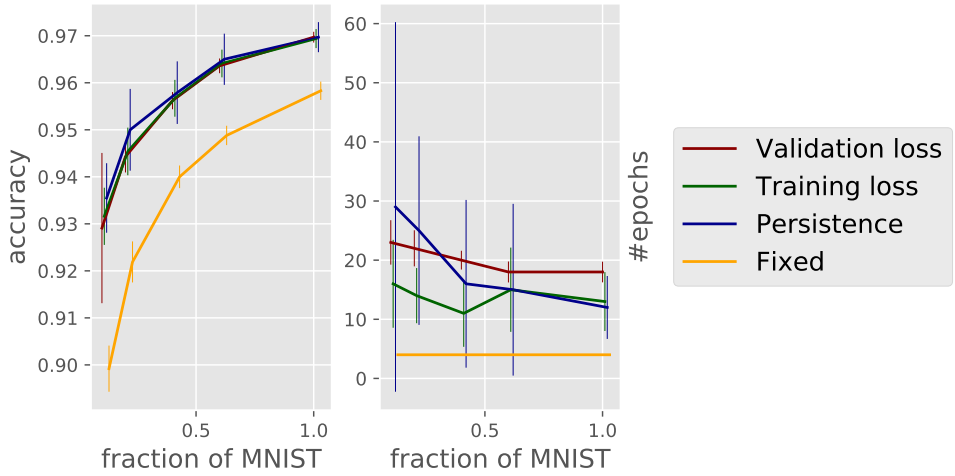

Figure A.4: A deep network is trained on a fraction of MNIST comparing four early stopping criteria with $g = 4$. On the left-hand side plot, the accuracies on the test set are evaluated. On the right-hand side, the number of trained epochs is shown. Both show averages and standard deviations over 20 runs.