Neural Persistence: A Complexity Measure for Deep Neural Networks using Algebraic Topology

Bastian Rieck[*,†], Matteo Togninalli[*,†], Christian Bock[*,†], Michael Moor[*], Max Horn[*], Thomas Gumbsch[*], Karsten Borgwardt[*]
[*]D-BSSE, Machine Learning and Computational Biology Lab, ETH Zurich, Switzerland; [†]Equal Contributions

Introduction

The practical successes of deep learning in various fields such as image processing, biomedicine, and language translation still outpace out theoretical understanding. Current approaches for improving theoretical and practical comprehension focus on interrogating networks with input data (e.g. feature visualizations for CNNs[1,2] or descriptive analyses based on information theory[3]).

In this work, we propose neural persistence, a complexity measure for neural network architectures based on topological data analysis that can be efficiently computed. To demonstrate the usefulness of our approach, we show that neural persistence reflects 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 achieving comparable accuracies as early stopping based on validation loss without the usage of a validation data set.

Background

Persistent Homology

The central object in algebraic topology is a simplicial complex K, e.g. an undirected weighted graph. Persistent homology captures topological features of K on different scales by defining a filtration on it. A filtration is a nested sequence of simplicial complexes (e.g. subgraphs K_m) representing the "growth" of K:

$$\emptyset = \mathrm{K}_0 \subseteq \mathrm{K}_1 \subseteq \cdots \subseteq \mathrm{K}_{m-1} \subseteq \mathrm{K}_m = \mathrm{K}$$

During this growth process, topological features can be *created* (a new connected component emerges) and *destroyed* (two connected components merge into one). The tuples of creation, and destruction times are captured as points (ϵ_c, ϵ_d) in a persistence diagram D. The persistence of a topological feature is then defined as

$$\operatorname{pers}(\epsilon_c, \epsilon_d) := |\epsilon_d - \epsilon_c|$$

Figure 1 illustrates the filtration process of a 2-dimensional point cloud.

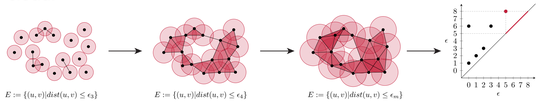


Fig. 1. The filtration process of a 2-dimensional point cloud and its persistence diagram. At each filtration step, the radius at which points can merge increases, leading to a decreasing number of connected components (e.g. 11 at ϵ_3 and 1 at ϵ_0).

Method Overview

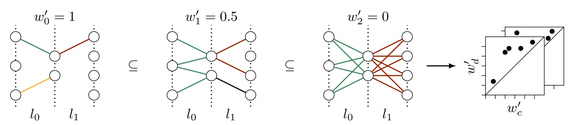


Fig. 2. Illustrating the neural persistence calculation of a network. Colours indicate connected components per layer. In contrast to figure 1, we reverse the filtration process by starting it with a threshold of 1 and decrease it during the filtration. As the threshold decreases, connectivity increases. Creation and destruction thresholds are collected in one persistence diagram per layer.

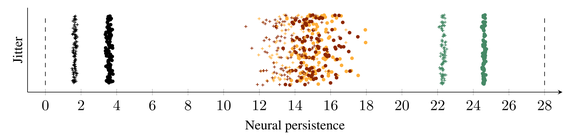


Fig. 2. Neural persistence values of trained perceptrons (green), diverging ones (yellow), random Gaussian matrices (red), and random uniform matrices (black). We performed 100 runs per category; dots indicate neural persistence while crosses indicate the predicted lower bound according to Theorem 2.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

Neural persistence (NP) captures neural network dynamics of the training process by exploiting both network structure and weight information using persistent homology.

We define NP on the k^{th} layer of a stratified graph (e.g. a feedforward neural network) $\mathrm{G}=(\mathrm{V}, \mathrm{E})$ satisfying $\mathrm{V}=\mathrm{V}_0 \sqcup \mathrm{V}_1 \sqcup \ldots$, such that if $\mathrm{u} \in \mathrm{V}_\mathrm{i}$, $\mathrm{v} \in \mathrm{V}_{\mathrm{i}+1}$, and $(\mathrm{u}, \mathrm{v}) \in \mathrm{E}$, we have $\mathrm{j} = \mathrm{i} + 1$. Given $\mathrm{k} \in \mathbb{N}$, the k^{th} layer of G is the unique subgraph

$$G_k := (V_k \sqcup V_{k+1}, E_k := E \cap \{V_k \times V_{k+1}\})$$

To simplify the comparison of different networks, we transform all network weights w of each training step such that $\mathrm{w'} \in [0,1]$ and sort them in non-ascending order. This permits us to define the **filtration** for the k^{th} layer of a neural network as:

$$G_k^{(i)} := (V_k \sqcup V_{k+1}, \{(u,v) \mid (u,v) \in E_k \wedge \varphi'(u,v) \geq w'_i\})$$

Our complexity measure is defined as the p-norm of the persistence diagram D_k that results from this filtration:

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

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[4] of topological features in a persistence diagram.

The calculation of NP is highly efficient: The filtration amounts to sorting all n weights of a network, which has **computational complexity** of $\mathrm{O}(\mathrm{n} \log \mathrm{n})$. Calculating persistent homology of this filtration can be realized using an algorithm based on union-find data structures[5] that exhibit a computational complexity of $\mathrm{O}(\mathrm{n} \cdot \alpha(\mathrm{n}))$, where $\alpha(\cdot)$ refers to the extremely slow-growing inverse of the Ackermann function[6].

To compare networks with different architectures (e.g. number and sizes of layers), bounds can be derived that allow us to normalize NP. Theorem 1 shows the upper bound which is used to define mean normalized NP, the measure that is used in our experiments:

Theorem 1. *Let $\varphi_k \colon E_k \to \mathcal{W}'$ denote the function that assigns each edge of G_k a transformed weight. The neural persistence $\mathrm{NP}(G_k)$ of the k^{th} layer satisfies*

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

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

Mean normalized NP of a *full* neural network is calculated by averaging normalized NP over all layers allowing us to side-step the problem of different layers having different scales:

Definition 1 (Mean normalized neural persistence)**.**

$$\widetilde{\mathrm{NP}}(G_k) := \mathrm{NP}(G_k) \cdot \mathrm{NP}(G_k^+)^{-1}$$
$$\overline{\mathrm{NP}}(G) := 1/l \cdot \sum_{k=0}^{l-1} \widetilde{\mathrm{NP}}(G_k)$$

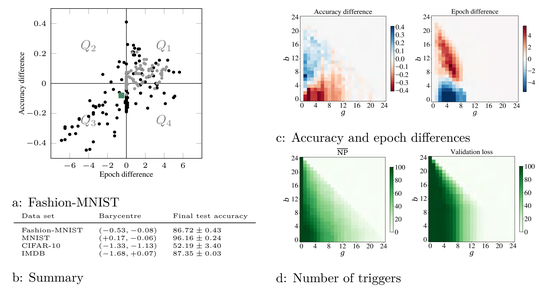
While Theorem 1 gives a lower and upper bound in a general setting, it is possible to obtain empirical bounds when we consider the tuples that result from the computation of a persistence diagram. Recall that our filtration ensures that the persistence diagram of a layer contains tuples of the form $(1, \mathrm{w}_\mathrm{i})$, with $\mathrm{w}_\mathrm{i} \in [0,1]$ 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 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*

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

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.

Results



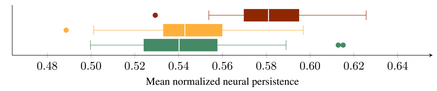
Data set	Barycentre	Final test accuracy
Fashion-MNIST	$(-0.53, -0.08)$	86.72 ± 0.43
MNIST	$(+0.17, -0.06)$	96.16 ± 0.24
CIFAR-10	$(-1.33, -1.13)$	52.19 ± 3.40
IMDB	$(-1.68, +0.07)$	87.35 ± 0.03

Early Stopping

Neural persistence can be used to evaluate the state of a training procedure and we evaluated its effectiveness as a stopping criterion by comparing it to validation loss across multiple comparable training runs and several patience *g* and burn-in *b* parameters. On average, **neural persistence stops earlier** with slightly worse accuracy (Table b). Moreover, the criterion stops reliably for a wide range of early stopping parameters.
The major benefit, however, is that **no validation data is necessary to prevent overfitting**.

Neural Persistence & Deep Learning Best Practices

We observe that networks trained with dropout yield higher normalized neural persistence compared to a regularly trained network. This can be explained because individual parts of the network are trained independently, resulting in a higher degree of per-layer redundancy.



References

1. Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, volume 8689 of Lecture Notes in Computer Science, pp. 818–833 Cham, Switzerland, 2014. Springer.
2. Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *Workshop Track of the International Conference on Learning Representations (ICLR)*, 2015.
3. Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015.
4. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.
5. Herbert Edelsbrunner, David Letscher, and Afra J. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.
6. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 3rd edition, 2009.