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 & 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 neural persistence to be a meaningful measure of structural complexity, it should increase as a neural network is learning. We evaluate this and other properties in Section 4.

Algorithm 1 provides pseudocode for 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 $\mathcal{O}\left(n \cdot \alpha\left(n\right)\right)$, 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://github.com/BorgwardtLab/Neural-Persistence`.

3.2 Properties of neural persistence

We elucidate properties about neural persistence to permit 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 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*

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

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

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

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

so 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*

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

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 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

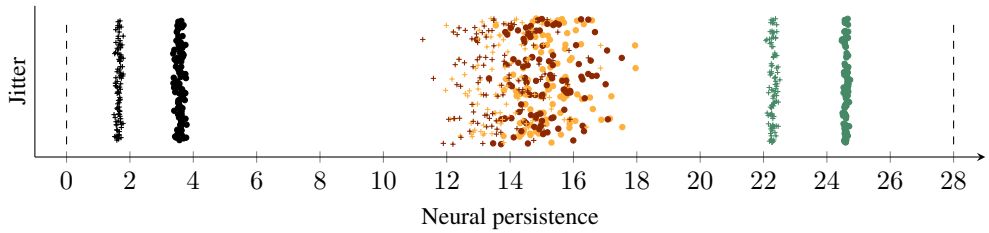

Figure 2: Neural persistence values of trained perceptrons (green), diverging ones (yellow), random Gaussian matrices (red), and random uniform matrices (black). We performed 100 runs per category; dots indicate neural persistence while crosses indicate the predicted lower bound according to Theorem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

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 Section A.1 in the appendix). For deeper networks, additional experiments discuss the relation between validation accuracy and neural persistence (Section A.5), the impact of different data distributions, as well as the variability of neural persistence for architectures of varying depth (Section A.6).

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 Table A.1 in the appendix for further details about the experiments.

4.1 Deep learning best practices in light of neural persistence

We compare the mean normalized neural persistence (see Definition 4) of a two-layer (with an architecture of $[650, 650]$) neural network to two models where batch normalization (Ioffe & 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., 2016). As individual parts of the network are trained independently,

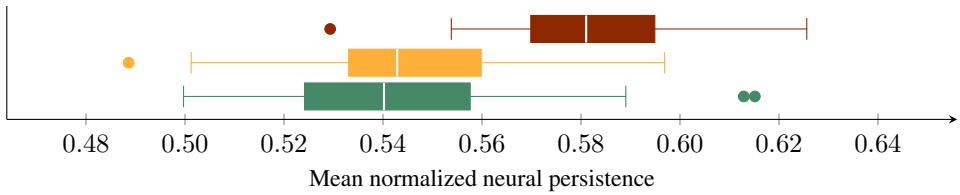

Figure 3: Comparison of mean normalized neural persistence for trained networks without modifications (green), with batch normalization (yellow), and with 50% of the neurons dropped out during training (red) for the 'MNIST' data set (50 runs per setting).