different groups was observed, validating the pruning method as a suitable way to generate normalized persistence diagrams to compare different models.

A slightly modified relevance function was used in filtrations by Watanabe and Yamana (2020) to perform edge pruning in neural networks. Using $R_{v,w} = \left|W_{i,j}^{(l+1)}\right| / \sum_{k\neq i} \left|W_{k,j}^{(l+1)}\right|$, Watanabe and Yamana proposed the following pruning algorithm: 1. Sort points (b, d) of one dimensional persistence diagrams by their value $b + d$ in ascending order; 2. For each point, choose a cycle representative c of the point (b, d); 3. Select the edges contained in the representatives in the previous step until you reach a desired number of edges to conserve; 4. Prune the edges not selected in the previous step. This method was shown to be competitive with respect to the global magnitude algorithm (Blalock et al., 2020) in terms of final accuracy of the pruned networks.

A similar approach to the one taken by Corneanu et al. (2020) and by Ballester et al. (2023b) was proposed by Barbara et al. (2024). Barbara et al. argued that the dataset has too much influence on the activations and thus their correlations do not capture all the relevant information about the neural network related with generalization. Instead, they proposed to analyze zeroth persistence diagrams $D(V_0(\mathrm{VR}(P, d)))$ induced by a subset P of the vertices of a neural network $\mathcal{N}$ and dissimilarities given by the Euclidean distance between scores based on weights associated to each vertex. The *score* associated to a node $v \in V(G(\mathcal{N}))$ in the network graph is the relevance value S_v given by

$$S_v = \sum_{(v,w)\in E(G(\mathcal{N}))} \pi_{w,v} \cdot \delta_w, \quad \pi_{w,v} = \frac{|W_{w,v}|}{\sum_{(u,w)\in E(G(\mathcal{N}))} |W_{w,u}|}, \quad \delta_w = \begin{cases} 1 & \text{if } w \in V_L, \\ S_w & \text{otherwise,} \end{cases}$$

where $W_{w,v}$ is the weight corresponding to the edge $(v, w) \in E(G(\mathcal{N}))$ and V_L is the set of neurons of the output layer. This score is the HVS score for a neuron (Yacoub and Bennani, 1997) which gives, for each neuron, a score of the neuron based on the magnitudes of the weights of its outcoming edges, that is related to the contribution of the neuron to the output value. The set P of points used to computed persistence diagrams is simply the 90% of neurons with highest relevance score.

To study the relationship of these diagrams with the generalization gap, Barbara et al. trained very simple FCFNNs with two hidden layers for a variety of datasets from the UCI Machine Learning repository (Kelly et al.). For the experiments, the generalization gap was compared with the average persistence value of the diagram generated at each iteration of the training procedure. Linear regression models were fitted with the generalization gaps and average persistences during training as dependent and independent variables, respectively. The R^2 values of these linear regressions were slightly greater than the ones for the same linear regression models using the persistence diagrams computed in Ballester et al. (2023a). However, the simplicity in the experiments performed, both in the experimental pipeline and in the networks used, makes that further experimentation is needed to evaluate if this approach generalizes to more complex scenarios and also improves the methods of Ballester et al. in them.

Weights layer by layer

Layerwise topological complexities have also been studied as a function of the weights. Rieck et al. (2019) proposed to study the so-called neural persistence. Given a non-output layer

l of a neural network $\mathcal{N}$, its *neural persistence* is defined as

$$\mathrm{NP}_l(\mathcal{N}) = \left(\sum_{(b,d)\in D} |d-b|^p \right)^{1/p},$$

where $D = D^w\left(\mathbb{V}_0\left(\mathrm{VR}^1(V(G_l), d_{\downarrow}^V)\right)\right)$ is the zeroth persistence diagram of the weighted complete bipartite graph G_l with vertices $V_l \sqcup V_{l+1}$ and weights given by $w_V(v) = w_{\max}$ and $w_E(\{v_l^i, v_{l+1}^j\}) = |W_{j,i}^{(l+1)}|/w_{\max}$, where $W_{j,i}^{(l+1)}$ is the weight associated to the edge connecting the vertices v_l^i and v_{l+1}^j, as in Figure 2, and $w_{\max} = \max_{l,i,j} |W_{i,j}^l|$ is the maximum weight in absolute value among all the weights of the network.

To compare different layers from the network, Rieck et al. proposed to normalize neural persistence by dividing it by an upper bound of the neural persistence,

$$\mathrm{NP}_l^+(\mathcal{N}) = w_{\max}^{-1}\left(\max_{i,j}\left|W_{i,j}^{(l+1)}\right| - \min_{i,j}\left|W_{i,j}^{(l+1)}\right|\right)(N_l - 1)^{1/p},$$

obtaining the normalized neural persistence $\widetilde{\mathrm{NP}}_l(\mathcal{N})$ for each layer l. Averaging normalized neural persistences over all the layers of the network, a global topological complexity measure $\overline{\mathrm{NP}}(\mathcal{N})$ is obtained.

For simple FCFNN networks trained on MNIST, neural persistence was able to distinguish clearly between properly and badly trained networks, for which the values of the neural persistence of properly trained ones were consistently higher than for their badly trained counterparts. Furthermore, it was observed that different regularization techniques augmented the mean neural persistence with respect to the values of regular trained networks, suggesting that for a fixed architecture, the higher the neural persistence, the better the generalization capabilities of the network. This was exploited as an early stopping criterion for training neural networks, where the training process is completed when the mean neural persistence $\overline{\mathrm{NP}}(\mathcal{N})$ stops increasing significantly. This early stopping criterion was found to be competitive with other early stopping criteria but without the need for a validation dataset, whose use is not always possible due to data scarcity.

Girrbach et al. deepened more into the properties of neural persistence. On the one hand, they discovered tighter bounds than those originally presented by Rieck et al. (2019) for the value of neural persistence. On the other hand, they also observed that there is a close relationship between the variance of the learned weights of deep learning models and the neural persistence, questioning the value of the latter with respect to this simpler measure to study the properties of the neural network, arguing that this variance may be similarly useful for the applications showcased by Rieck et al.

Activations whose dissimilarities depend on the weights

Both weights and activations can be used together to study the topology of neural networks. In particular, they can be used to inspect differences in the internal workings of neural networks for different input values. Given an input sample x and a neural network $\mathcal{N}$, Gebhart et al. (2019) proposed to analyze the zeroth persistence module $\mathbb{V}_0(\mathrm{VR}^1(V(G_x), d_{\downarrow}^0))$ for G_x the undirected weighted graph induced by the directed neural network graph $G(\mathcal{N})$ with weights given by the formula $w_E(\{v_l^i, v_{l+1}^j\}) = |W_{j,i}^{(l+1)} \phi_{\mathcal{N}}^{(l)}(x)_i|$. The weight function captures how activations are distributed through the neural network, as in the previous works. However, in this case, activations are weighted by the weights of the neural network, which do not necessarily depend on the example x, and which have an effect of normalization on the influence of the input. Each persistence module is intended to capture information of the network under the influence of input x. In this way, the differences between persistence modules for the same network and different inputs can be used to gain insight into the dynamics and representations used by the neural network to compute its function.

To measure the differences between persistence modules coming from the same network $\mathcal{N}$ and different inputs x, x' quantitatively, Gebhart et al. proposed to calculate a dissimilarity based on the differences between the cycle representatives of the points of the two persistence diagrams $D_x = D(\mathbb{V}_0(\mathrm{VR}^1(V(G_x), d_{\downarrow}^0)))$ and $D_{x'} = D(\mathbb{V}_0(\mathrm{VR}^1(V(G_{x'}), d_{\downarrow}^0)))$, respectively. More precisely, let $\{\alpha_i\}_{i=1}^{|D_x|}$ and $\{\beta_j\}_{j=1}^{|D_{x'}|}$ be the representatives of the points of D_x and $D_{x'}$, respectively. Each representative cycle α_i and β_j is associated with a subgraph of the graph G_x and $G_{x'}$, respectively, denoted by T_i and T_j'. Build two vectors v_x and v_x' with as many components as edges there are in the union of graphs $\left(\cup_{i=1}^{|D_x|} T_i\right) \cup \left(\cup_{j=1}^{|D_{x'}|} T_j'\right)$, respectively, where v_x and v_y have one in the components corresponding to the edges that come from the union of graphs $\left(\cup_{i=1}^{|D_x|} T_i\right)$ and $\left(\cup_{j=1}^{|D_{x'}|} T_j'\right)$, respectively, and a zero otherwise. Then, the dissimilarity between the persistence modules generated from x and x' is defined as a weighed version of the Hamming distance between v_x and $v_{x'}$ using the persistence of sthe cycle representatives associated with the edges corresponding to the different components of the vectors as weights.

The utility and relevance of such persistence modules and their dissimilarity to the study of neural networks was proven in several ways for simple scenarios involving the MNIST, FashionMNIST, and CIFAR-10 datasets and three simple convolutional architectures, including a variant of AlexNet (Krizhevsky et al., 2012) for the CIFAR-10 dataset. For example, persistence modules showed excellent classification performance as input to support vector machines (SVMs) using kernel dissimilarity. In practice, the inputs were transformed into their persistence modules for a fixed trained network $\mathcal{N}$, from which the classification was performed. This approach was tested for some selected trained neural networks, for which the trained SVMs were accurate, even for some adversarial examples. Furthermore, dissimilarity between persistence modules was found to resemble distances between the original images in the input space. This suggests that much of the information used by the neural network to classify the examples is contained in the topology as extracted by the persistence modules and that this information is different enough to understand the differences between the behavior of the neural network with respect to different classes.

The aforementioned persistence modules were further studied to detect adversarial examples by Goibert et al. (2022). The hypothesis is that only a small set of edges within the neural networks are used for inference of non-adversarial inputs and that, for adversarial examples, the number of edges used for inference is larger. The idea behind the hypothesis is that adversarial examples attack input-output edge paths with underused edges of the neural network to, with imperceptible modifications of the inputs, completely change the output. Ideally, these changes in the activations of the neurons included in the underused paths make a change in the structure of the activations of the neural network and thus in