7 layers distributed following a bottleneck architecture with ReLU activation functions except for the output one, topological complexities were observed to increase in the last layers on average and only decreased in the first layers. Observations on these two different architectures question the previous results by Naitzat et al. and motivate further research on the topic.

Another promising approach, deeply related to the study of the decision regions, is the analysis of the topology of the activations of the output layer. The fundamental hypothesis is similar to what was found in Naitzat et al. (2020): the easier the topology of the last layer, the more robust, and thus the better, the neural network. In this regard, Akai et al. (2021) studied the zero and one-dimensional Vietoris–Rips persistence diagrams of the last-layer activations, taken as in the previous articles, using the usual Euclidean distance. Although they did not prove the main hypothesis and could not relate network performance with output layer topology, it is a first step towards this direction.

Intuitively, the topology of the activations for inputs of the same class must be similar. Based on this hypothesis, Zhao and Zhang (2022) propose a way to measure the *quality* of convolutional filters for individual channels of convolutional layers in neural networks for classification problems. In this work, the inputs are squared images, and we assume that the widths and heights of convolutional layer outputs are also equal. To compare the topology of the activations produced by the convolutional filters of interest, undirected weighted fully connected graphs are built. For a layer l, channel $c \in [c^{(l-1)}]$, and input sample x, the filter graph $C_x^{l,c}$ is generated such that $V(C_x^{l,c}) = [h]$ and

$$w_E(\{i,j\}) = \max\left(\phi_{\mathcal{N}}^{(l)}(x)_{i,j,c},\, \phi_{\mathcal{N}}^{(i)}(x)_{j,i,c}\right),$$

for $i, j \in [c^{(l-1)}]$, where $\phi_{\mathcal{N}}^{(l)}(x)_{j,i,c}$ is the (i,j,c) output value of $\mathcal{N}$ for the l-th convolutional layer given x. For each of these graphs, Zhao and Zhang computed the infimum of the support of the associated Betti curves coming from the persistence module $\mathbb{V}_1(\mathrm{VR}^1(V(C_x^{l,c}), d_{\downarrow}^0))$, which is directly related to the time in which the first *non-trivial cycle*, i.e., not given by a combination of triangles, appears in the filtration. We denote this minimum by $b_1^{\inf}(x,l,c)$. The values $b_1^{\inf}(x,l,c)$ induce discrete random variables $B_1(l,c,y)$ for each label y of the classification problem with sample space Ω_y the subset of samples from the training dataset $\mathcal{D}_{\text{train}}$ with common label y. These random variables have probability distributions

$$P_{1,l,c,y}(n) = \frac{b_n}{|\Omega_y|}, \qquad b_n = \sum_{x\in\Omega_y} \mathbb{1}_{\left\{x' \,:\, b_1^{\inf}(x',l,c)=n\right\}}(x),$$

that capture information about the similarity of the topology of the activations given by the convolutional filters of interest in the data distributions of the different labels of the classification problem. This similarity can be further quantified using the entropies of the probability distributions, given by

$$H'_{1,l,c,y} = -\sum_{n=0}^{\infty} P_{1,l,c,y}(n) \log P_{1,l,c,y}(n). \tag{15}$$

The values given by Equation (15) are called *feature entropies*, and are the topological summaries proposed by Zhao and Zhang to measure the quality of convolutional filters.

Note that the infimum value $b_1^{\inf}$ may not exist in some cases. For layers and channels in which this happens many times, the entropy approaches zero, although entropy is not really informative as it is affected by the non-existence of infimum values. In such cases, we modify entropy as

$$H_{1,l,c,y} = \begin{cases} H'_{1,l,c,y} & \text{if } \varepsilon_{1,l,c,y} \geq p, \\ (1 - \varepsilon_{1,l,c,y}) \log |\Omega_y| & \text{otherwise,} \end{cases}$$

where $\varepsilon_{1,l,c,y}$ is the percentage of images in class y having birth times and p the minimum percentage of images we admit to use the real entropy, in the article, $p = 0.1$.

Zhao and Zhang demonstrated the effectivity of the entropy measure to obtain information on the *quality* of the convolutions and of the entire neural network for VGG-16 models trained on the ImageNet dataset. They observed that, for well-trained neural networks, the feature entropy continually decreased as the layers went deeper, while for random weights this decreasement is absent. On the other hand, they observed that, during training, the feature entropies of the last convolutional layer decreased, and were highly correlated with the evolution of the cross entropy training loss, suggesting that feature entropies are good indicators of the generalization of networks.

Feature entropies also were invariant to weights reescaling, a desirable property to measure the quality of the convolution operations, as reescalings of weights have no substantial impact on the network performance in general. Additionally, randomness of the weights was also detected by comparing the feature entropies of trained and randomly initialized networks. Finally, for the last convolutional layers, models with better generalization were connected to low feature entropies.

Weights in the complete neural network graph

Recall that the input values influence the output of a neural network via the different input-output paths available in the neural network graph. The influence of each path is characterized by the magnitudes of the weights associated with the edges in the path, which modify the magnitudes of the activations, and thus their relevance, at each step. Watanabe and Yamana (2022b) propose to build neural network graph filtrations taking into account the influence of the different paths in the network.

Let $\mathcal{N}$ be a neural network, and $v_l^i, v_{l+1}^j \in V(G(\mathcal{N}))$ be two connected neurons in the directed graph $G(\mathcal{N})$. Formally, the relevance of the directed edge (v_l^i, v_{l+1}^j) connecting v_l^i and v_{l+1}^j in the output calculation is defined to be the linearly rectified weight of the edge normalized by the sum of the linearly rectified weights of edges pointing to v_{l+1}^j, that is,

$$R_{v,w} = \frac{\operatorname{ReLU}\big(W_{i,j}^{(l+1)}\big)}{\sum_{k \neq i} \operatorname{ReLU}\big(W_{k,j}^{(l+1)}\big)}.$$

Then, the influence from a vertex v to a vertex w is characterized as the maximum product of the relevances of the edges among all the paths between v and w whenever $v \neq w$, and 1 otherwise, i.e.,

$$\tilde{R}_{v,w} = \begin{cases} \max_{(v,p_1,\ldots,p_n,w) \in P_{v,w}} R_{v,p_0} \left(\prod_{i=1}^{n-1} R_{p_i,p_{i+1}} \right) R_{p_n,w} & \text{if } v \neq w, \\ 1 & \text{otherwise,} \end{cases}$$

where $P_{v,w}$ is the set of all paths starting at v and ending at w in $G(\mathcal{N})$.

The influences between vertices of a neural network induce a dissimilarity function over the neurons of the network, which can be used to build filtrations. However, to preserve information about edge directions, Watanabe and Yamana propose an alternative way to build filtrations $(K_i)_{i=1}^n$ from a monotonically decreasing sequence of indices $(t_i)_{i=1}^n$ given by

$$K_i^p = K_{t_i}^p = \begin{cases} V(G(\mathcal{N})) & \text{if } p = 0, \\ \{\{v_{k_0}\ldots, v_{k_p}\} : \tilde{R}_{v_{k_i}, v_{k_j}} \geq t_i \text{ for all } k_i > k_j\} & \text{if } p \geq 1, \end{cases}$$

where p indicates the dimensions of the simplices of K_i^p, and the vertices $V(G(\mathcal{N}))$ are ordered in such a way that if the layer number of v_i is lower or equal than the layer number of v_j, then $i \geq j$.

The filtration $(K_i)_{i=1}^n$ captures the influence of the different paths between neurons in the neural network graph and can be easily extended to a persistence module where the simplicial complex assigned to the time $t \in \mathbb{R}$ is $K_{\lfloor t \rfloor}$ for $0 \leq t \leq n$, K_0 for $t < 0$, and K_n for $t > n$. Preliminary results about simple networks trained on MNIST and CIFAR-10 suggest that the distribution of points of one-dimensional persistence diagrams computed from persistence modules coming from the previous filtration are correlated with several properties of the network and the dataset, such as problem difficulty or network expressivity. For example, the appearance of points near the diagonal of the persistence diagrams was associated with a shortage of data of specific labels in the training dataset. In addition, persistence diagrams seemed to be robust with respect to the initial weight values of the neural network weights, yielding each architecture similar persistence diagrams for different initializations after training. However, a more thorough study of such persistence modules and their associated persistence diagrams is needed to understand their relationship with the generalization capabilities of a neural network, as Watanabe and Yamana remark.

An extension of the relevance quantity for convolution and pooling operations leads to persistence modules better suited to analyze convolutional neural networks. This extension was presented in Watanabe and Yamana (2022a) to study the overfitting of convolutional neural networks. Watanabe and Yamana observed that, in simple scenarios, the distribution of the points in the one-dimensional persistence diagram was correlated with the overfitting of the network, as measured by the difference in train and test accuracies. Specifically, for each given architecture, the number of points near the diagonal increased according to the increase in the dropout rate of the network used in their experiments, which was correlated with the overfitting of the network. In addition, a high total number of points in the persistence diagram was correlated with less underfitting in neural networks.

The observed correlation between points of persistence diagrams and generalization of neural networks make persistence diagrams potential tools for comparing neural networks according to their generalization capabilities. However, straightforward persistence diagrams were found not to be entirely suitable for comparing models with different architectures. To address this difficulty, Watanabe and Yamana proposed a novel approach: pruning the networks of interest using a magnitude-based strategy (Blalock et al., 2020) before generating persistence diagrams to compare them. After selecting several groups of neural networks to be compared, an overall high correlation between the number of points near the diagonal of normalized persistence diagrams and overfitting of the networks of the