4 Challenges, future directions, and conclusions

In this survey we have seen many examples of how topological data analysis, and particularly (persistent) homology and Mapper, can be applied to study the properties of neural networks. In Section 3.1, we saw how to compute two homology groups that yielded different information about a neural network only by taking its graph without weights. However, we noticed that the ranks of those homology groups were invariant to many important structural properties of the neural network, such as the order of the layers. Also, their values were simply a combination of the width and the depth of the graph, not capturing much information about the network. To obtain better insight into the properties of neural networks, more fine-grained homology theories for directed graphs are needed. A promising line of work would be to study the persistent path homology (Chowdhury and Mémoli, 2018) of graphs augmented with the weights associated to each edge, as in Section 3.3.

In Section 3.2, we saw how topology was useful in recovering the *topology* of decision regions and boundaries, with applications especially in model selection. Furthermore, we saw that GTDA, an evolution of the Mapper algorithm for graphs as input, was successful in analyzing patterns in output spaces of neural networks, leading to a better understanding of misclassification errors in several datasets.

Thanks to the differentiability theory of persistence diagrams, we saw how to *improve* decision regions by *simplifying* them. In addition, we discussed how the topology was useful for capturing the quality —measured in different ways— of generative models. Particularly interesting is the study of the disentanglement of generative models, since being able to decouple the different sources of variation could lead to a better control and quality of the networks' outputs. We find that refining the article by Zhou et al. (2021) and using its measures to regularize generative neural networks could be an interesting approach to improving neural network design using TDA.

In Section 3.3 we analyzed the methods studying the neural network parameters and activations. We split the section into the articles using Mapper and (persistent) homology, because of the differences in their approaches. In the first case, we found both studies of the parameters and of the activations. In the first case, we saw how Carlsson and Brüel Gabrielsson found very interesting connections between the topologies of the space of natural images and the space of weights of some layers in convolutional networks, among many other interesting topology configurations of the convolutional weights. Also, we saw how Gabella studied the Mapper graphs of the evolution of the weights during training, finding a very interesting pattern in the weight distribution of one of the architectures tried in his experiments: the weights seemed to be distributed in a *surface!* A closer look at the topology of the weight evolution for a higher variety of architectures must be taken to realize if the weights share common patterns in similar problems. This could potentially enable leveraging this structure, as proposed in Carlsson and Brüel Gabrielsson (2020), for instance. For activations, we found that the general approach proposed by TopoAct (Rathore et al., 2021) was extremely useful for understanding the internal behavior of neural networks, with the many ramifications presented in the survey.

For (persistent) homology papers, we could broadly classify the approaches depending on either the set of neurons/edges analyzed (layer by layer versus all the neurons), or the dissimilarity strategy (weights versus distances between activations or versus a combination of weights and activations).

In many cases, there was a connection between the different topological summaries extracted from Vietoris–Rips persistence diagrams and the properties of neural networks, especially with their generalization capacities. However, Girrbach et al. showed that most of the information extracted by one of the most influential articles in this section could be captured by taking simpler, non-topological summaries of the neural network. Although this does not mean that TDA does not provide unique information about the internal workings of the neural networks, we think that more ablation studies must be performed in the studies claiming the utility of TDA for the analysis of neural networks, as almost all the work performed in this area is experimental, and, usually, there is no theory supporting the hypotheses connecting topological summaries and neural network properties.

Another interesting discussion in this subsection is the opposition of views on the evolution of the topology of the data through the different layers. In most works, such as the one by Naitzat et al. (2020), it is stated and verified that the topology of the data is *simplified* by the action of the layers. However, the work by Wheeler et al. (2021) arrived to the opposite conclusion. This contradiction may have occurred due to the simplicity of the experiments performed in all of the experiments and supports the necessity of performing more diverse and complete experiments.

The previous point leads to one of the fundamental drawbacks of the section: most of the experiments are performed on classical CNN and FCFNN architectures and do not say anything about more modern architectures, like transformers. To be useful, TDA must be applied to state-of-the-art architectures, not only as a proof of concept on *simple* neural networks. This is a fundamental step for gaining the trust of the non-TDA deep learning community in TDA methods. Also, since many people working on this area are mathematicians, it would be desirable that TDA methods be accompanied by theoretical results, as in Birdal et al. (2021).

In Section 3.4, we saw how TDA was used to study the loss function and training weights. In this case, we observed that persistence diagrams, which come originally from topology, were used to compute fractal dimensions, that belong to the geometry realm. This is an exciting result because it means that persistent homology can be used not only to infer the topology of the data, but also to infer geometrical properties. This is further studied by Andreeva et al. (2023), and we think that seeing persistent homology as a tool that also extracts local information from the data could be useful in performing new work to analyze the structures of neural networks.

Probably the most critical drawback of topological data analysis for neural networks is the high computational complexity in time and memory of computing invariants, like persistence diagrams, of persistence modules. For dimension zero, algorithms based on the minimum spanning tree (Chazelle, 2000) or the single linkage algorithm (Sibson, 1973), have time complexities $\mathcal{O}(e \cdot \alpha(e, n))$ and $\mathcal{O}(n^2)$, respectively, where α is the very slow growing functional inverse of the Ackermann function (Tarjan, 1975) and e is the number of edges of the simplicial complex, whose cardinality is $\mathcal{O}(n^2)$. Also, for single-linkage clustering, the memory complexity is linear, that is, $\mathcal{O}(n)$. This makes these algorithms suitable for some of the applications for which the number of points is relatively small, such as the analysis of weights or activations layer by layer. However, for bigger point clouds, such as in the problems in which we use the whole set of neurons or weights, these algorithms become infeasible for modern and big neural networks. The situation is worse for dimensions greater than or equal to one, where typical persistence diagram computation algorithms have a time and memory complexity of $\mathcal{O}(n^w)$ and $\mathcal{O}(n^2)$, respectively, where w is the matrix multiplication exponent (currently $w < 2.4$) and n is the number of simplices generated through the filtration of simplicial complexes (Birdal et al., 2021).

Computational problems motivate the development of more suitable algorithms to compute persistence diagrams or alternative invariants that also capture information of neural networks. Although powerful tools for computing Vietoris–Rips persistence diagrams are available, such as Ripser (Bauer, 2021), Ripser++ (Zhang et al., 2020), Gudhi (The GUDHI Project, 2015), SLINK (Sibson, 1973), algorithms that parallelize computations are of special interest due to the increasing availability of powerful hardware and software methods to perform distributed computing. Also, approximating known filtrations by means of new filtrations, as in (Sheehy, 2013), or by means of machine learning models, as in Montufar et al. (2020).

As we have seen in this survey, there are many different ways to produce meaningful filtrations for neural networks in all the domains we reviewed. For this reason, it would be desirable to capture the joint information of some of them at the same time. Also, for some specific problems, like analyzing the evolution of the data distribution through the layers, filtrations by distances alone may not be good enough, as they do not consider important data properties such as the density of points in the sample taken. Luckily, multiparameter persistence (Botnan and Lesnick, 2023) deals with this kind of problem setting, allowing to extract information from non-linear filtrations, that is, with more than one varying parameter. Unluckily, multiparameter persistence does not have a canonical, easy-to-use and computationally efficient representation like persistence diagrams for one-dimensional persistence. However, there are already works proposing a rich body of useful, and even differentiable, representations for multiparameter persistence; see, for example Loiseaux et al. (2023a,b); Xin et al. (2023). We see multiparameter persistence as one of the main lines of work not only for the analysis of deep neural networks, but also for the whole machine learning community, as multiparameter persistence provides sharper ways to extract information from data. However, we are far from having usable representations for real neural network use cases due to the huge quantity of points in real datasets and neurons in architectures. For this reason, further fundamental research on the topic is needed before we can grasp its advantages in the deep learning community. Further work in this direction could be based on three basic pillars: 1. Ease of use of multiparameter representations that can be stored in any conventional computer, since representations must be intuitive for the average machine learning researcher; 2. Efficiency of computation; 3. Differentiability of representations with respect to the point clouds, to allow regularization in learning tasks.

In conclusion, we have seen that the use of TDA applied to studying neural networks is an exciting path that can be useful in many scenarios. TDA, although computationally expensive, is a tool that has been shown to be connected with many interesting properties of neural networks such as generalization in classification problems or entanglement and latent space quality in generative models. Moreover, we have seen that there are many ways in which TDA can be used in applied scenarios, converting TDA into an essential tool for deep learning practitioners.