Topological Uncertainty: Monitoring Trained Neural Networks through Persistence of Activation Graphs

Théo Lacombe[1*], **Yuichi Ike**[3], **Mathieu Carrière**[2],
Frédéric Chazal[1], **Marc Glisse**[1], **Yuhei Umeda**[3]
[1]Université Paris-Saclay, CNRS, Inria, Laboratoire de Mathématiques d'Orsay, France
[2]Université Côte d'Azur, Inria, France
[3]Fujitsu Ltd., Japan
{theo.lacombe, mathieu.carriere, frederic.chazal, marc.glisse}@inria.fr,
{ike.yuichi, umeda.yuhei}@fujitsu.com

Abstract

Although neural networks are capable of reaching astonishing performances on a wide variety of contexts, properly training networks on complicated tasks requires expertise and can be expensive from a computational perspective. In industrial applications, data coming from an open-world setting might widely differ from the benchmark datasets on which a network was trained. Being able to monitor the presence of such variations without retraining the network is of crucial importance. In this article, we develop a method to monitor trained neural networks based on the topological properties of their activation graphs. To each new observation, we assign a *Topological Uncertainty*, a score that aims to assess the reliability of the predictions by investigating the whole network instead of its final layer only, as typically done by practitioners. Our approach entirely works at a post-training level and does not require any assumption on the network architecture, optimization scheme, nor the use of data augmentation or auxiliary datasets; and can be faithfully applied on a large range of network architectures and data types. We showcase experimentally the potential of Topological Uncertainty in the context of trained network selection, Out-Of-Distribution detection, and shift-detection, both on synthetic and real datasets of images and graphs.

1 Introduction

Over the last decade, Deep Learning (DL) has become the most popular approach to tackle complex machine learning tasks, opening the door to a broad range of industrial applications. Despite its undeniable strengths, monitoring the behavior of deep Neural Networks (NN) in real-life applications can be challenging. The more complex the architectures become, the stronger the predictive strengths of the networks, but the looser our grasp on their behaviors and weaknesses.

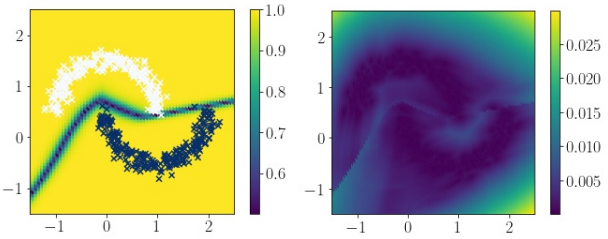

Figure 1: *(Left)* Confidence (maximum in the final soft-max layer) of the predictions made by a neural network trained on the two-moons dataset (white and black crosses) over the plane. Away from the thin classification boundary (confidence value close to 0.5), the network tends to produce *over-confident predictions* (value close to 1), even for points that are far away from the training data. *(Right)* Topological distance between all activation graphs and the activation graphs computed on the train set (higher means less confident).

Training a neural network requires task-specific expertise, is time consuming, and requires the use of high-end expensive hardware. With the rise of companies providing model marketplaces (*e.g.*, [Kumar *et al.*, 2020, Table 1] or `tensorflow-hub`), it is now common that users only have access to fixed trained neural networks, with few information on the training process, and would rather avoid training the networks again.

In practice, a network can perform well on a given learning problem—in the sense that it can achieve high accuracy on the training and test sets—, but lack reliability when used in real-life applications thereafter. One can think of, among other examples, *adversarial attacks*, that are misinterpreted by neural networks with high confidence levels [Nguyen *et al.*, 2015; Akhtar and Mian, 2018; Biggio and Roli, 2018], *calibration* issues leading to under- or over-confident predictions [Guo *et al.*, 2017; Ovadia *et al.*, 2019; Hein *et al.*, 2019], lack of *robustness* to corruption or perturbation of the input data [Hendrycks and Dietterich, 2019].

Related Work

Many techniques have been proposed to improve or monitor the behavior of NN deployed in real-life applications. Most require specific actions taken during the training phase

*Contact Author

of the network; for instance via *data augmentation* [Shorten and Khoshgoftaar, 2019]), the use of large auxiliary datasets[1] [Hendrycks *et al.*, 2019], modifications of the network architecture [DeVries and Taylor, 2018] or its objective function [Atzmon *et al.*, 2019; Van Amersfoort *et al.*, 2020], or using several networks at once to produce predictions [Lakshminarayanan *et al.*, 2017]. The approach we introduce in this article focuses on acting at a post-training level only, and since our goal is to benchmark it against the use of confidence alone in a similar setting, we use, in our experimental results, the baseline introduced in [Hendrycks and Gimpel, 2017], which proposes to monitor NNs based on their confidences and on the idea that low-confidence predictions may account for anomalies.

Using topological quantities to investigate NN properties has experienced a growth of interest recently (see for instance [Guss and Salakhutdinov, 2018; Carlsson and Gabrielsson, 2020]). In [Gebhart and Schrater, 2017; Gebhart *et al.*, 2019], the authors introduce the notion of activation graphs and showcase their use in the context of adversarial attacks. We mix this idea with [Rieck *et al.*, 2019], which proposes to investigate topological properties of NN layer-wise[2]. A topological score based on an estimation of the NN decision boundary has been proposed in [Ramamurthy *et al.*, 2019] to perform trained network selection, an idea we adapt in Subsection 4.1.

Contributions

In this work, we propose a new approach to monitor trained NNs by leveraging the topological properties of *activation graphs*. Our main goal is to showcase the potential benefits of investigating network predictions through the lens of the whole network instead of looking at its *confidence* encoded by its final layer only as usually done in practice.

To that aim, we introduce *Topological Uncertainty* (TU), a simple topological quantity that, for a given NN and a new observation, encodes how the network "reacts" to this observation, and whether this reaction is similar to the one on training data. Our approach does not require any assumption on the network training phase nor the type of input data, and can thus be deployed in a wide variety of settings. Furthermore, it only relies on computating maximum spanning trees (MST), leading to a simple and efficient implementation.

Experimentally, we show that TU can be used to monitor trained NNs and detect Out-of-Distribution (OOD) or shifted samples when deployed in real-life applications. Our results suggest that TU can drastically improve on a standard baseline based on the network confidence in different situations. Our implementation will be incorporated in the `Gudhi` library[3].

[1]In particular, the `80 million tiny images dataset` [Torralba *et al.*, 2008], used as an auxiliary dataset in state-of-the-art OOD detection techniques [Chen *et al.*, 2020], has been withdrawn for ethical concerns. This situation illustrates unexpected limitations when relying on such datasets to calibrate neural networks.

[2]A detailed comparison is deferred to the supplementary material

[3]https://gudhi.inria.fr/

2 Background

2.1 Neural Networks

For the sake of clarity, we restrict our presentation to *sequential* neural networks, although our approach (detailed in Section 3) can be generalized to more general architectures, *e.g.*, recurrent neural networks. We also restrict to classification tasks; let d denote the dimension of the input space and K be the number of classes. A (sequential) neural network (NN) learns a function $F\colon \mathbb{R}^d \to \mathbb{R}^K$ that can be written as $F = f_L \circ \cdots \circ f_1$, where the $(f_\ell)_{\ell=1}^L$ are elementary blocks defined for $\ell = 1, \dots, L-1$ as

$$x_{\ell+1} = f_\ell(x_\ell) = \sigma_\ell(W_\ell \cdot x_\ell + b_\ell),$$

with $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$, $b_\ell \in \mathbb{R}^{d_{\ell+1}}$, and $(\sigma_\ell)_\ell$ are activation maps[4], *e.g.*, $\sigma_\ell = \mathrm{ReLU}\colon x \mapsto \max\{x, 0\}$. In classification tasks, the final activation $f_L = \sigma_L$ is usually taken to be the soft-max function, so that the output $x_L = F(x)$ can be understood as a probability distribution on $\{1, \dots, K\}$ whose entries $(F(x)_k)_k$ indicate the likelihood that x belongs to class k. The predicted class is thus $\arg\max_k\{F(x)_k\}$, while the *confidence* that the network has in its prediction is $\max_k\{F(x)_k\} \in [0, 1]$. Given a *training set* of observations and labels $(X^{\text{train}}, Y^{\text{train}}) = (x_i, y_i)_{i=1}^{N_{\text{train}}}$ distributed according to some (unknown) joint law $(\mathcal{X}, \mathcal{Y})$, the network parameters W_ℓ, b_ℓ are optimized to minimize the loss $\sum \mathcal{L}(F(x_i), y_i)$ for some loss function $\mathcal{L}$ (*e.g.*, the categorical cross-entropy). The (training) accuracy of F is defined as $\frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} 1_{\arg\max(F(x_i))=y_i}$.

2.2 Activation Graphs and Topological Descriptors

Activation graphs. Let us consider a neural network F and two layers of size d_ℓ and $d_{\ell+1}$ respectively, connected by a matrix $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$. One can build a bipartite graph G_ℓ whose vertex set is $V_\ell \sqcup V_{\ell+1}$ with $|V_\ell| = d_\ell$ and $|V_{\ell+1}| = d_{\ell+1}$, and edge set is $E_\ell = V_\ell \times V_{\ell+1}$. Following [Gebhart *et al.*, 2019], given an instance $x \in \mathbb{R}^d$, one can associate to each edge $(i, j) \in E_\ell$ the weight $|W_\ell(i, j) \cdot x_\ell(i)|$, where $x_\ell(i)$ (resp. $W_\ell(i, j)$) denotes the i-th coordinate of $x_\ell \in \mathbb{R}^{d_\ell}$ (resp. entry (i, j) of $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$). Intuitively, the quantity $|W_\ell(i, j) \cdot x_\ell(i)|$ encodes how much the observation x "activates" the connection between the i-th unit of the ℓ-th layer and the j-th unit of the $(\ell+1)$-th layer of F. In this way, we obtain a sequence of bipartite graphs $(G_\ell(x, F))_\ell$ called the *activation graphs* of the pair (x, F), whose vertices are $V_\ell \sqcup V_{\ell+1}$ and edges weights are given by the aforementioned formula.

Maximum spanning trees and persistence diagrams. To summarize the information contained in these possibly large graphs in a quantitative way, we rely on topological descriptors called *persistence diagrams*, coming from the Topological Data Analysis (TDA) literature. A formal introduction to TDA is not required in this work (we refer to the supplementary material for a more general introduction): in our specific

[4]Note that this formalism encompasses both fully-connected and convolutional layers that are routinely used by practitioners.

case, persistence diagrams can be directly defined as the distribution of weights of a *maximum spanning tree* (MST). We recall that given a connected graph G with $N+1$ vertices, a MST is a connected acyclic sub-graph of G sharing the same set of vertices such that the sum of the N edge weights is maximal among such sub-graphs. MST can be computed efficiently, namely in quasilinear time with respect to the number of edges in G. Given the ordered weights $w_1 \geq \cdots \geq w_N$ of a MST built on top of a graph G, its persistence diagram is the one-dimensional probability distribution

$$\mu(G) := \frac{1}{N}\sum_{i=1}^{N} \delta_{w_i},$$

where δ_{w_i} denotes the Dirac mass at $w_i \in \mathbb{R}$. See Figure 2 for an illustration.

Comparing and averaging persistence diagrams. The standard way to compare persistence diagrams relies on *optimal partial matching metrics*. The choice of such metrics is motivated by stability theorems that, in our context, imply that the map $x \mapsto \mu(G_\ell(x,F))$ is Lipschitz with Lipschitz constant that only depends on the network architecture and weights[5] (and not on properties of the distribution of $x \sim \mathcal{X}$ for instance). The computation of these metrics is in general challenging [Kerber *et al.*, 2017]. However, in our specific setting, the distance $\mathrm{Dist}(\mu,\nu)$ between two diagrams $\mu = \frac{1}{N}\sum_{i=1}^{N}\delta_{w_i}$ and $\nu = \frac{1}{N}\sum_{j=1}^{N}\delta_{w'_j}$ can be simply obtained by computing a 1D-optimal matching[5], which in turn only requires to match points in increasing order, leading to the simple formula

$$\mathrm{Dist}(\mu,\nu)^2 = \frac{1}{N}\sum_{i=1}^{N} |w_i - w'_i|^2,$$

where $w_1 \geq w_2 \geq \cdots \geq w_N$ and $w'_1 \geq w'_2 \geq \cdots \geq w'_N$. With this metric comes a notion of *average persistence diagram*, called a *Fréchet mean* [Turner *et al.*, 2014]: a Fréchet mean $\overline{\mu}$ of a set of M diagrams $\mu_1, \ldots, \mu_M$ is a minimizer of $\nu \mapsto \sum_{m=1}^{M} \mathrm{Dist}(\mu_m,\nu)^2$, which in our context simply reads

$$\overline{\mu} := \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{w}_i},$$

where $\bar{w}_i = \frac{1}{M}\sum_{m=1}^{M} w_i^{(m)}$ and $w_i^{(m)}$ denotes the i-th point of μ_m. The Fréchet mean provides a geometric way to concisely summarize the information contained in a set of persistence diagrams.

3 Topological Uncertainty (TU)

Building on the material introduced in Section 2, we propose the following pipeline, which is summarized in Figure 2. Given a trained network $F\colon \mathbb{R}^d \to \mathbb{R}^K$ and an observation x, we build a sequence of activation graphs $G_1(x,F), \ldots, G_{L-1}(x,F)$. We then compute a MST of each $G_\ell(x,F)$, which in turn induces a persistence diagram $D_\ell(x,F) := \mu(G_\ell(x,F))$. Assuming F was trained on a set X^{train}, one can store the corresponding sequence of diagrams $(D_\ell(x^{\text{train}},F))_\ell$ for each $x^{\text{train}} \in X^{\text{train}}$. These diagrams summarize how the network is activated by the training data. Thus, given a new observation x with $\arg\max(F(x)) = k$, one can compute the sequence $(D_\ell(x,F))_\ell$ and then the quantity

$$\min_{\substack{x^{\text{train}} \in X^{\text{train}},\\ \arg\max(F(x^{\text{train}}))=k}} \mathrm{Dist}\left(D_\ell(x,F), D_\ell(x^{\text{train}},F)\right),$$

that is, comparing $D_\ell(x,F)$ to the diagrams of training observations that share the same predicted label than x. Figure 1 (right) shows how this quantity evolves over the plane, and how, contrary to the network confidence (left), it allows one to detect instances that are far away from the training distribution.

Since storing the whole set of training diagrams for each class and each layer $\{D_\ell(x^{\text{train}},F) \,:\, \arg\max(F(x^{\text{train}})) = k\}$ might be inefficient in practice, we propose to summarize these sets through their respective Fréchet means $\overline{D^{\text{train}}_{\ell,k}}$. For a new observation $x \in \mathbb{R}^d$, let $k(x) = \arg\max(F(x))$ be its predicted class, and $(D_\ell(x,F))_\ell$ the corresponding persistence diagrams. The *Topological Uncertainty* of x is defined to be

$$\mathrm{TU}(x,F) := \frac{1}{L}\sum_{\ell=1}^{L} \mathrm{Dist}\left(D_\ell(x,F), \overline{D^{\text{train}}_{\ell,k(x)}}\right), \quad (1)$$

which is the average distance over layers between the persistence diagrams of the activation graphs of (x,F) and the average diagrams stored from the training set. Having a low TU suggests that x activates the network F in a similar way to the points in X^{train} whose class predicted by F was the same as x. Conversely, an observation with a high TU (although being possibly classified with a high confidence) is likely to account for an OOD sample as it activates the network in an unusual manner.

Remarks. Our definition of activation graphs differs from the one introduced in [Gebhart *et al.*, 2019], as we build one activation graph for each layer, instead of a single, possibly very large, graph on the whole network. Note also that the definition of TU can be declined in a variety of ways. First, one does not necessarily need to work with all layers $1 \leq \ell \leq L$, but can only consider a subset of those. Similarly, one can estimate Fréchet means $\overline{D^{\text{train}}_{\ell,k}}$ using only a subset of the training data. These techniques might be of interest when dealing with large datasets and deep networks. One could also replace the Fréchet mean $\overline{D^{\text{train}}_{\ell,k}}$ by some other diagram of interest; in particular, using the empty diagram instead allows us to retrieve a quantity analog to the *Neural persistence* introduced in [Rieck *et al.*, 2019]. On an other note, there are other methods to build persistence diagrams on top of activation graphs that may lead to richer topological descriptors, but our construction has the advantage of returning diagrams supported on the real line (instead of the plane as it usually occurs in TDA) with fixed number of points, which dramatically simplifies the computation of distances and Fréchet means, and makes the process efficient practically.

[5] We refer to the supplementary material for details and proofs.

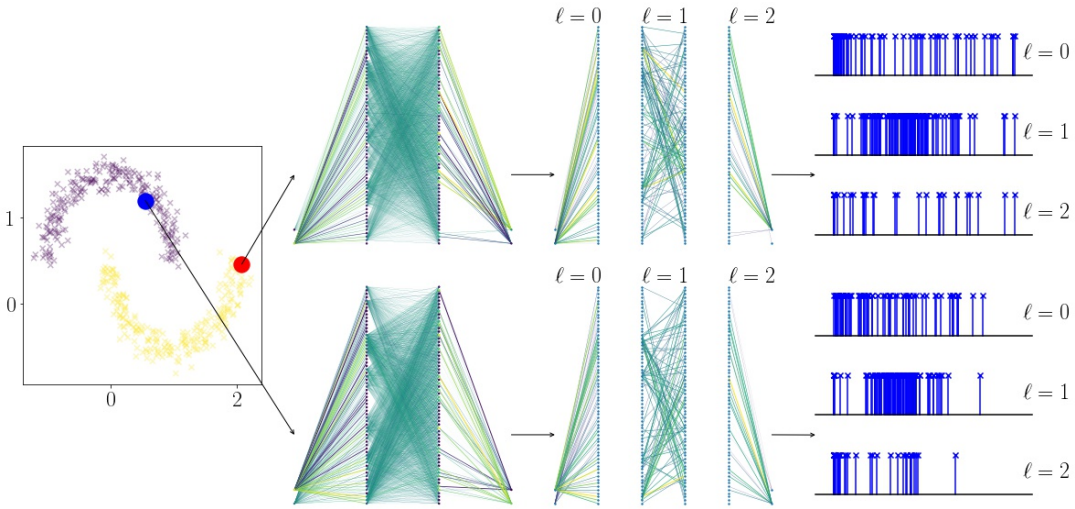

Figure 2: Pipeline presented in this article: Each observation activates the network with a weight $|W_\ell(i,j) \cdot x_\ell(i)|$ on the edge connecting the i-th unit in the ℓ-th layer to the j-th unit of the $(\ell+1)$-th layer. A maximum spanning tree is then computed for each layer ℓ of the network, whose distribution of weights provides a corresponding *persistence diagram*. On this example, the network used is a simple network with two hidden-layers of 64 units each with ReLU activation, each layer is fully-connected (dense matrix).

4 Experiments

This section showcases the use of TU in different contexts: trained network selection (§4.1), monitoring of trained networks that achieve large train and test accuracies but have not been tweaked to be robust to Out-Of-Distribution observations (§4.2) or distribution shift (§4.3).

Datasets and experimental setting. We use standard, publicly available, datasets of graphs and images. `MNIST`, `Fashion-MNIST`, `CIFAR-10`, `SVHN`, `DTD` are datasets of images, while `MUTAG` and `COX2` are datasets of graphs coming from a chemical framework. We also build two OOD image datasets, `Gaussian` and `Uniform`, by randomly sampling pixel values following a Gaussian distribution (resp. uniform on the unit cube). A detailed report of datasets and experimental settings (data preprocessing, network architectures and training parameters, etc.) can be found in the supplementary material.

4.1 Trained Network Selection for Unlabeled Data

In this subsection, we showcase our method in the context of trained network selection through an experiment proposed in [Ramamurthy *et al.*, 2019, §4.3.2]. Given a dataset with 10 classes (here, `MNIST` or `Fashion-MNIST`), we train 45 NNs on the binary classification problems i *vs.* j for each pair of classes (i,j) with $i > j$, and store the average persistence diagrams of the activation graphs for the different layers and classes as explained in Section 3. These networks are denoted by F_{ij} in the following, and consistently reach high accuracies on their respective training and test sets given the simplicity of the considered tasks. Then, for each pair of classes $k_1 > k_2$, we sample a set of new observations X_{k_1,k_2} made of 200 instances sampled from the *test* set of the initial dataset (in particular, these observations have not been seen during the training of any of the $(F_{ij})_{ij}$) whose labels are k_1 and k_2. Assume now that k_1, k_2 and the labels of our new observations are unknown. The goal is to select a network that is likely to perform well on X_{k_1,k_2} among the $(F_{ij})_{ij}$. To that aim, we compute a *score* for each pairing $(k_1,k_2) \leftrightarrow (i,j)$, which is defined as the average TU (see Eq. (1)) when feeding F_{ij} with X_{k_1,k_2}. A low score between (i,j) and (k_1,k_2) suggests that X_{k_1,k_2} activates F_{ij} in a "known" way, thus that F_{ij} is likely to be a relevant classifier for X_{k_1,k_2}, while a high score suggests that the data in X_{k_1,k_2} is likely to be different from the one on which $F_{i,j}$ was trained, making it a less relevant choice. Figure 3 plots the couple (scores, accuracies) obtained on `MNIST` when taking $(k_1,k_2) = (1,0)$, that is, we feed networks that have been trained to classify between i and j handwritten digits with images representing 0 and 1 digits. Not surprisingly, $F_{1,0}$ achieves both a small TU (thus would be selected by our method) and a high accuracy. On the other hand, using the model $F_{7,6}$ on $X_{1,0}$ leads to the highest score and a low accuracy.

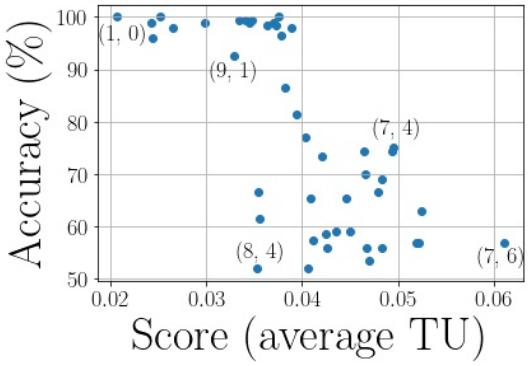

Figure 3: Score and accuracies obtained on 45 models trained on the MNIST dataset when fed with a set $X_{1,0}$ of images representing 0 and 1 digits. The point annotations refer to the values (i,j) on which the network was initially trained. For instance, a network trained on 7 and 4 has a score of 0.49 and an accuracy of $\sim 75\%$ on $X_{1,0}$.

To quantitatively evaluate the benefit of using our score to select a model, we use the the metric proposed by [Ramamurthy *et al.*, 2019]: the difference between the mean accuracy obtained using the 5 models that have the lowest scores and the 5 models that have the highest scores. We obtain a $+14.8\%$-increase on `MNIST` and a $+12.6\%$-increase on `Fashion-MNIST`. These positive values indicate that, on average, using low TU as a criterion helps to select a better model to classify our new set of observations. As a comparison, using the average network confidence as a score leads to $+6.6\%$ on `MNIST` and $+4.7\%$ on `Fashion-MNIST`, indicating that using (high) confidence to select a model would be less relevant than using TU, on average. See the supplementary material for a complementary discussion.

Remark. Our setting differs from the one of [Ramamurthy *et al.*, 2019]: in the latter work, the method requires to have access to true labels on the set of new observations X_{k_1,k_2} (which we do not), but do not need to evaluate $F_{ij}(x),\ x \in X_{k_1,k_2}$ on each model F_{ij} (which we do). To that respect, we stress the complementary aspect of both methods.

4.2 Detection of Out-Of-Distribution Samples

The following experiment illustrates the behavior of TU when a trained network faces Out-Of-Distribution (OOD) observations, that is, observations that are not distributed according to the training distribution. To demonstrate the flexibility of our approach, we present the experiment in the context of graph classification, relying on the `COX2` and the `MUTAG` graph datasets. Complementary results on image datasets can be found in Table 1 and in the supplementary material. Working with these datasets is motivated by the possibility of using simple networks while achieving reasonably high accuracies which are near state-of-the-art on these sets. To train and evaluate our networks, we extract 40 spectral features from graphs—thus representing a graph by a vector in $\mathbb{R}^{40}$—following a procedure proposed in [Carrière *et al.*, 2020]. See the supplementary material for details.

For both datasets, we build a set of 100 *fake graphs* in the following way. Let $\mathcal{N}$ and $\mathcal{M}$ be the distributions of number of vertices and number of edges respectively in a given dataset (*e.g.*, graphs in `MUTAG` have on average 17.9 ± 4.6 vertices and 19.8 ± 5.7 edges). Fake graphs are sampled as Erdős-Renyi graphs of parameters $(n, m/n^2)$, with $n \sim \mathcal{N}$, $m \sim \mathcal{M}$, thus by construction fake graphs have (on average) the same number of vertices and edges as graphs from the training dataset. These sets are referred to as `Fake-MUTAG` and `Fake-COX2`, respectively.

Now, given a network $F_{\texttt{MUTAG}}$ trained on `MUTAG` (resp. $F_{\texttt{COX2}}$ trained on `COX2`), we store the average persistence diagrams of each classes. It allows us to compute the corresponding distribution of TUs $\mathcal{T}^{\text{train}} = \{\mathrm{TU}(x, F_{\texttt{MUTAG}}) : x \in \texttt{MUTAG}\}$ (resp. $\mathrm{TU}(x, F_{\texttt{COX2}})$). Similarly, we evaluate the TUs of graphs from the `Fake-MUTAG` (resp. `Fake-COX2`) and from `COX2` (resp. `MUTAG`), see Figure 4 ((a,c), respectively). As expected, the TUs of training inputs $\mathcal{T}^{\text{train}}$ are concentrated around low values. Conversely, the TUs of OOD graphs (both from the Fake dataset and the second graph dataset) are significantly higher. Despite important differences in terms of TU, the network still shows confidence near 100% ((b,d) plots) for OOD datasets, making this quantity impractical for detecting OOD samples.

		Baseline		Topological Uncertainty	
Training data	OOD data	FPR ↓	AUC ↑	FPR ↓	AUC ↑
MUTAG	Fake-MUTAG	98.4	1.7	**0.0**	**99.8**
	COX2	93.0	31	**0.0**	**100.0**
COX2	Fake-COX2	91.2	1.4	**0.0**	**99.9**
	MUTAG	91.2	1.1	**0.0**	**100.0**
CIFAR-10	FMNIST	93.6	54.9	**65.6**	**86.4**
	MNIST	94.7	58.3	**25.4**	**94.7**
	SVHN	90.6	27.6	**83.6**	**65.8**
	DTD	90.9	32.6	90.3	**57.3**
	Uniform	91.5	31.8	**59.1**	**80.2**
	Gaussian	91.0	27.2	**18.8**	**88.2**

Table 1: Comparison between the baseline confidence-based OOD-detector and our TU-based classifier on graph datasets (first two rows) and on image datasets for a network trained on `CIFAR-10`. FPR is given at TPR 95%. ↑: higher is better, ↓: lower is better.

To quantify this intuition, we propose a simple OOD detector. Let F be a network trained on `MUTAG` or `COX2`, with distribution of training TUs $\mathcal{T}^{\text{train}}$. A new observation x is classified as an OOD sample if $\mathrm{TU}(x, F)$ is larger than the q-th quantile of $\mathcal{T}^{\text{train}}$. This classifier can be evaluated with standard metrics used in OOD detection experiments: the *False Positive Rate at 95% of True Positive Rate* (FPR at 95%TPR), and the Area Under the ROC Curve (AUC). We compare our approach with the baseline introduced in [Hendrycks and Gimpel, 2017] based on confidence only: a point is classified as an OOD sample if its confidence is *lower* than the q-th quantile of the distribution of confidences of training samples. As recorded in Table 1, this baseline performs poorly on these graph datasets, which is explained by the fact that (perhaps surprisingly) the assumption of *loc. cit.* that training samples tend to be assigned a larger confidence than OOD-ones is not satisfied in this experiment. In the third row of this Table, we provide similar results for a network trained on `CIFAR-10` using other image datasets as OOD sets. Although in this setting TU is not as efficient as it is on graph datasets, it still improves on the baseline reliably.

4.3 Sensitivity to Shifts in Sample Distribution

This last experimental subsection is dedicated to *distribution shift*. Distribution shifts share similar ideas with OOD-detection, in the sense that the network will be confronted to samples that are not following the training distribution. However, the difference lies in the fact that these new observations are shifted instances of observations that would be sampled with respect to the training distribution. In particular, shifted instances still have an underlying label that one may hope to recover. Formally, given a training distribution $\mathcal{X}$ and a parametric family of shifts $(s_\gamma)_\gamma$ with the convention that $s_0 = \mathrm{id}$, a shifted sample with level of shift γ is a sample $s_\gamma(x_1), \ldots, s_\gamma(x_N)$, where $x_1, \ldots, x_N \sim \mathcal{X}$ with underlying labels $y_1, \ldots, y_N$. For instance, given an image, one can apply a *corruption shift* of parameter $\gamma = n \in \mathbb{N}$ where $s_n(x)$ consists of randomly switching n pixels of the image x ($x_{ij} \mapsto 1 - x_{ij}$). See the top row of Figure 5.

Ideally, one would hope a trained network F to be robust to shifts, that is $\arg\max(F(x)) = \arg\max(F(s_\gamma(x)))$. However, since the map $x \mapsto s_\gamma(x)$ cannot be inverted in general, one cannot realistically expect robustness to hold for high lev-

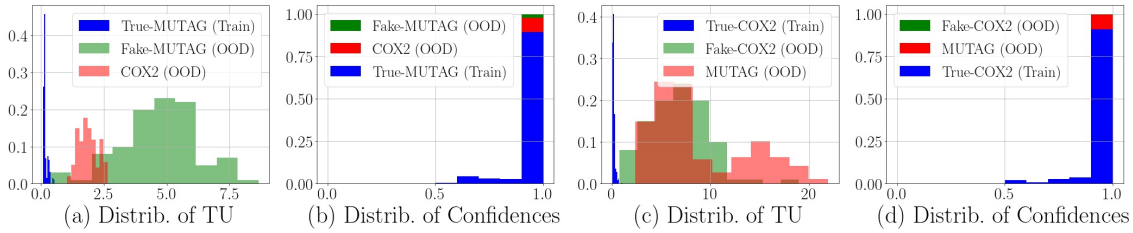

Figure 4: *(a)* Distributions of Topological Uncertainties (TU) of a network trained on the `MUTAG` dataset. Blue distribution corresponds to $\mathcal{T}^{\text{train}}$. Green and red distributions correspond to topological uncertainties of observations coming from `Fake-MUTAG` and `COX2` datasets respectively. *(b)* Distributions of network confidences ($\max\{F(x)_k\}$). The network makes overconfident predictions, especially on OOD datasets that are classified almost systematically with a confidence of 1. *(c,d)* Similar plots for the `COX2` dataset.

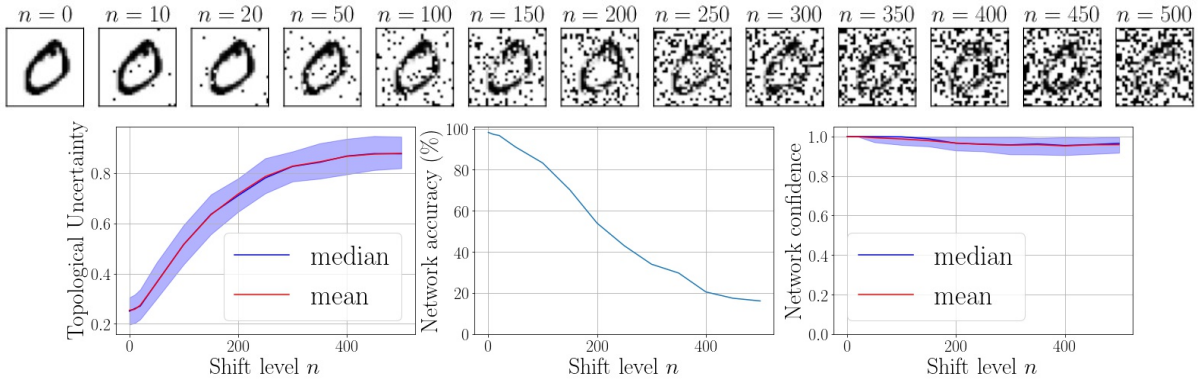

Figure 5: *(Top row)* A 0 digit from the `MNIST` dataset exposed to increasing level of shift (pixel corruption). *(Bottom row), (Left).* The TU (with 0.1 and 0.9 quantiles) of corrupted inputs in the `MNIST` dataset with respect to the corruption level n. *(middle)* The accuracy of the network on these data (that is, the proportion of observations that are still correctly classified). *(right)* The confidence of the network in its predictions. Although the accuracy is dropping significantly, the network remains overall extremely confident in its predictions.

els of shift. Here, we illustrate how TU can be used as a way to monitor the presence of shifts that would lead to a dramatic diminution of the network accuracy in situations where the network confidence would be helpless.

For this, we train a network on `MNIST` and, following the methodology presented in Section 3, store the corresponding average persistence diagrams for the 10 classes appearing in the training set. We then expose a batch of 1000 observations from the test set containing 100 instances of each class (that have thus not been seen by the network during the training phase) to the corruption shifts with various shift levels $n \in \{0, 10, 20, 50, 100, 150, \ldots, 500\}$. For each shift level, we evaluate the distributions of TUs and confidences attributed by the network to each sample, along with the accuracy of the network over the batch. As illustrated in the second row of Figure 5, as the batch shifts, the TU increases and the accuracy drops. However, the network confidence remains very close to 1, making this indicator unable to account for the shift. In practice, one can monitor a network by routinely evaluating the distribution of TUs of a new batch (*e.g.*, daily recorded data). A sudden change in this 1D distribution is likely to reflect a shift in the distribution of observations that may itself lead to a drop in accuracy (or the apparition of OOD samples as illustrated in Subsection 4.2).

We end this subsection by stressing that the empirical relation we observe between the TU and the network accuracy cannot be guaranteed without further assumption on the law $(\mathcal{X}, \mathcal{Y})$. It however occurs consistently in our experiments (see the supplementary material for complementary experiments involving different types of shift and network architectures). Studying the theoretical behavior of the TU and activation graphs in general will be the focus of further works.

5 Conclusion and Perspectives

Monitoring trained neural networks deployed in practical applications is of major importance and is challenging when facing samples coming from a distribution that differs from the training one. While previous works focus on improving the behavior of the network confidence, in this article we propose to investigate the whole network instead of restricting to its final layer. By considering a network as a sequence of bipartite graphs on top of which we extract topological features, we introduce the Topological Uncertainty, a tool to compactly quantify if a new observation activates the network in the same way as training samples did. This notion can be adapted to a wide range of networks and is independent from the way the network was trained. We illustrate numerically how it can be used to monitor networks and how it turns out to be a strong alternative to network confidence on these tasks. Our implementation will be integrated to the `Gudhi` library.

We believe that this work will motivate further developments involving Topological Uncertainty, and more generally activation graphs, when it comes to understand and monitor neural networks. In particular, most techniques introduced in recent years to improve confidence-based descriptors may be declined to be used with Topological Uncertainty. These trails of research will be investigated in future work.

References

[Akhtar and Mian, 2018] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[Atzmon *et al.*, 2019] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. In *NeurIPS*, pages 2034–2043, 2019.

[Biggio and Roli, 2018] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[Carlsson and Gabrielsson, 2020] Gunnar Carlsson and Rickard Brüel Gabrielsson. Topological approaches to deep learning. In *Topological Data Analysis*, pages 119–146. Springer, 2020.

[Carrière *et al.*, 2020] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: a neural network layer for persistence diagrams and new graph topological signatures. In *AISTATS*, pages 2786–2796. PMLR, 2020.

[Chen *et al.*, 2020] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Robust out-of-distribution detection via informative outlier mining. *arXiv preprint arXiv:2006.15207*, 2020.

[DeVries and Taylor, 2018] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[Gebhart and Schrater, 2017] Thomas Gebhart and Paul Schrater. Adversary detection in neural networks via persistent homology. *arXiv preprint arXiv:1711.10056*, 2017.

[Gebhart *et al.*, 2019] Thomas Gebhart, Paul Schrater, and Alan Hylton. Characterizing the shape of activation space in deep neural networks. In *2019 18th IEEE ICMLA*, pages 1537–1542. IEEE, 2019.

[Guo *et al.*, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, pages 1321–1330, 2017.

[Guss and Salakhutdinov, 2018] William H Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv preprint arXiv:1802.04443*, 2018.

[Hein *et al.*, 2019] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[Hendrycks and Dietterich, 2019] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[Hendrycks and Gimpel, 2017] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*, 2017.

[Hendrycks *et al.*, 2019] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *ICLR*, 2019.

[Kerber *et al.*, 2017] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22:1–20, 2017.

[Kumar *et al.*, 2020] Abhishek Kumar, Benjamin Finley, Tristan Braud, Sasu Tarkoma, and Pan Hui. Marketplace for ai models. *arXiv preprint arXiv:2003.01593*, 2020.

[Lakshminarayanan *et al.*, 2017] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 30:6402–6413, 2017.

[Nguyen *et al.*, 2015] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[Ovadia *et al.*, 2019] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, pages 13991–14002, 2019.

[Ramamurthy *et al.*, 2019] Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In *ICML*, pages 5351–5360. PMLR, 2019.

[Rieck *et al.*, 2019] Bastian Alexander Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *ICLR*. OpenReview, 2019.

[Shorten and Khoshgoftaar, 2019] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[Torralba *et al.*, 2008] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[Turner *et al.*, 2014] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52(1):44–70, 2014.

[Van Amersfoort *et al.*, 2020] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9690–9700. PMLR, 13–18 Jul 2020.