Homological Convolutional Neural Networks

Antonio Briola antonio.briola.20@ucl.ac.uk
Yuanrong Wang yuanrong.wang.20@ucl.ac.uk
Silvia Bartolucci s.bartolucci@ucl.ac.uk
Tomaso Aste t.aste@ucl.ac.uk
Department of Computer Science, University College London, London, WC1E 6BT, UK

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

Deep learning methods have demonstrated outstanding performances on classification and regression tasks on homogeneous data types (e.g., image, audio, and text data). However, tabular data still pose a challenge, with classic machine learning approaches being often computationally cheaper and equally effective than increasingly complex deep learning architectures. The challenge arises from the fact that, in tabular data, the correlation among features is weaker than the one from spatial or semantic relationships in images or natural language, and the dependency structures need to be modeled without any prior information. In this work, we propose a novel deep learning architecture that exploits the data structural organization through topologically constrained network representations to gain relational information from sparse tabular inputs. The resulting model leverages the power of convolution and is centered on a limited number of concepts from network topology to guarantee: (i) a data-centric and deterministic building pipeline; (ii) a high level of interpretability over the inference process; and (iii) an adequate room for scalability. We test our model on 18 benchmark datasets against 5 classic machine learning and 3 deep learning models, demonstrating that our approach reaches state-of-the-art performances on these challenging datasets. The code to reproduce all our experiments is provided at https://github.com/FinancialComputingUCL/HomologicalCNN.

Keywords: Topological Deep Learning, Tabular Learning, Networks, Complex Systems

1. Introduction

We are experiencing a tremendous and inexorable progress in the field of deep learning. Such a progress has been catalyzed by the availability of increasing computational resources and always larger datasets. The areas of success of deep learning are heterogeneous. However, the three application domains where superior performances have been detected are the ones involving the usage of images (He et al., 2015; Pak and Kim, 2017), audio (Purwins et al., 2019; Bose and Tripathy, 2020) and text (Lai et al., 2015; Chowdhary and Chowdhary, 2020; Zhang and Li, 2021). Despite their inherent diversity, these data types share a fundamental characteristic: they exhibit homogeneity, with notable inter-feature correlations and evident spatial or semantic relationships. On the contrary, tabular data represent the "unconquered castle" of deep neural network models (Kadra et al., 2021b). They are heterogeneous data types and present a mixture of continuous, categorical, and ordinal values, which can be either independent or correlated. They are characterized by the absence of any inherent positional information, and tabular models have to handle features from multiple discrete and continuous distributions. However, tabular data are the most common data format and

© 2023 A. Briola, Y. Wang, S. Bartolucci & T. Aste.

are ubiquitous in many critical applications, such as medicine (Ulmer et al., 2020; Somani et al., 2021), finance (Sachan et al., 2020; Ohana et al., 2021), recommendation systems (Zhang et al., 2019a, 2021), cybersecurity (Buczak and Guven, 2015; Rawat et al., 2019), and anomaly detection (Pang et al., 2022; Wang et al., 2022) – to mention a few. During the last decade, traditional machine learning methods dominated tabular data modeling, and, nowadays, tree ensemble algorithms (i.e. XGBoost, LightGBM, CatBoost) are the recommended option to solve real-life problems of this kind (Friedman, 2001; Prokhorenkova et al., 2018; Shwartz-Ziv and Armon, 2022).

In the current paper, we introduce a novel deep learning architecture for tabular numerical data classification and we name it "Homological Convolutional Neural Network" (`HCNN`). We exploit a class of information filtering networks (Barfuss et al., 2016; Briola et al., 2022; Briola and Aste, 2022, 2023; Vidal-Tomás et al., 2023; Wang et al., 2023b), namely the Triangulated Maximally Filtered Graph (Massara et al., 2017), to model the inner sparsity of tabular data and obtain a geometrical organization of input features. Emerging data relationships are hence studied at different granularity levels to capture both simplicial and homological structures through the usage of Convolutional Neural Networks (CNNs). Compared to state-of-the-art (SOTA) machine learning alternatives (Friedman, 2001; Chen and Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018), our method (i) maintains an equivalent level of explainability; (ii) has a comparatively lower level of computational complexity; and (iii) can be scaled to a higher number of learning tasks (e.g. time series forecasting) without structural changes. Compared to its SOTA deep-learning alternatives (Arik and Pfister, 2021; Badirli et al., 2020; Hazimeh et al., 2020; Huang et al., 2020; Klambauer et al., 2017; Kontschieder et al., 2015; Popov et al., 2019; Song et al., 2019; Somepalli et al., 2021; Beutel et al., 2018; Wang et al., 2019; Kadra et al., 2021a,b; Shavitt and Segal, 2018; Baosenguo, 2021), our method (i) is data-centric (i.e. the architecture depends on the data defining the system under analysis); (ii) presents an algorithmic data-driven building pipeline; and (iii) has a lower complexity, replacing complex architectural modules (e.g. attention-based mechanisms) with elementary computational units (e.g. convolutional layers). We provide a comparison between `HCNN`s, simple-to-advanced machine learning algorithms and SOTA deep tabular architectures using a heterogeneous battery of small-to-medium sized numerical benchmark datasets. We observe that `HCNN` always ties SOTA performances on the proposed tasks, providing, at the same time, structural and computational advantages.

2. Data and Methods

2.1. Data

To provide a fair comparison between `HCNN` and SOTA models, we use a collection of 18 tabular numerical datasets from the open-source "OpenML-CC18" benchmark suite (Bischl et al., 2017). Following the selection criteria in (Hollmann et al., 2022), all the datasets contain up to 2 000 samples, 100 features, and 10 classes. A deep overview on the properties of this first set of data is provided in Appendix A. Following (Grinsztajn et al., 2022), we focus on small datasets because of two main reasons: (i) small datasets are often encountered in real-world applications (Dua et al., 2017); and (ii) existing deep learning methods are limited in this domain. It is worth noticing that, differently from other deep learning

architectures (Hollmann et al., 2022; Arik and Pfister, 2021), the applicability of `HCNN`s is not limited to small tabular data problems and can easily scale to medium-to-large problems. To provide evidence of this, we use a collection of 9 numerical tabular datasets from the "OpenML tabular benchmark numerical classification" suite (Grinsztajn et al., 2022). All these datasets violate at least one of the selection criteria in (Hollmann et al., 2022) (i.e. they are characterized by a number of samples $> 2\,000$ or they are characterized by a number of features > 100). A deep overview on the properties of this second set of data is provided in Appendix A.

2.2. Information Filtering Networks

The `HCNN`'s building process is entirely centered on the structural organization of data emerging from the underlying sparse network representation. The choice of the network representation is not binding even if limited to the family of simplicial complexes (Torres and Bianconi, 2020; Salnikov et al., 2018). In this paper, we exploit the power of a class of information filtering networks (IFNs) (Mantegna, 1999; Aste et al., 2005; Barfuss et al., 2016; Massara et al., 2017; Tumminello et al., 2005), namely the Triangulated Maximally Filtered Graph (TMFG) (Massara et al., 2017), to model the inner sparsity of tabular data and obtain a structural organization of input features. IFNs are an effective tool to represent and model dependency structures among variables characterizing complex systems while imposing topological constraints (e.g. being a tree or a planar graph) and optimizing specific global properties (e.g. the likelihood) (Aste, 2022). Starting from a system characterized by n features and T samples, arranged in a matrix $\mathbf{X}$, this methodology builds a $n \times n$ similarity matrix $\hat{\mathbf{C}}$ which is filtered to obtain a sparse adjacency matrix $\boldsymbol{A}$ retaining only the most structurally significant relationships among variables. Working with numerical-only tabular data, in the current paper, $\hat{\mathbf{C}}$ corresponds to a matrix of squared correlation coefficients. To improve the robustness of the correlation similarity measure, in line with the work by (Tumminello et al., 2007), we use the bootstrapping approach (Efron et al., 1996). This technique requires to build a number r of replicas X_i^*, $i \in 1, \ldots, r$ of the data matrix $\mathbf{X}$. Each replica X_i^* is built by randomly selecting T rows from the matrix $\mathbf{X}$ allowing for repetitions. For each replica X_i^*, the correlation matrix $\hat{\mathbf{C}}_i^*$ is then computed. We highlight that (i) the bootstrap approach does not require the knowledge of the data distribution; and (ii) it is particularly useful to deal with high dimensional systems where it is difficult to infer the joint probability distribution from data. Once obtained replicas-dependent correlation matrices, we treat them in two different ways:

- We compute $\hat{\mathbf{C}}$ as the entry-wise mean of correlation matrices $\hat{\mathbf{C}}^*_{i\in 1,\ldots,r}$, and we construct a TMFG (see Appendix B) by using it.
- Based on each replica-dependent correlation matrix $\hat{\mathbf{C}}_i^*$, we compute a TMFG$_i^*$ (see Appendix B) and we obtain the final, filtered, TMFG by taking only the links that appear in all the TMFG* with a frequency higher than a specified threshold.

In the rest of the paper, we will refer to the first configuration as `MeanSimMatrix` and to the second one as `BootstrapNet`. These two approaches lead to widely different results. In the former case, the final TMFG will be a sparse, connected graph that necessarily maintains all the topological characterization of the family of IFNs it belongs to (i.e. planarity

(Tumminello et al., 2005) and chordality (Massara et al., 2017)). In the latter case, instead, there will be no guarantee on the connectedness of the graph. Indeed, the chosen threshold could lead to disconnected components and to the removal of edges assuring the graph's chordality.

2.3. Homological Convolutional Neural Networks

The main idea behind IFNs is to explicitly model higher-order sub-structures, which are crucial for the representation of the underlying system's interactions. Based on this, in a recent work (Wang et al., 2023a), the authors propose a simple higher-order representation which (i) starts from a layered representation (i.e. the Hasse diagram); (ii) explicitly takes into account higher-order sub-structures (i.e. simplices) and their higher-order interconnections (i.e. homological priors); (iii) and converts this representation into a stand-alone computational unit named "Homological Neural Network" (`HNN`). Despite the undeniable advantages deriving from this sparse architecture (see Appendix C), results suggest that the authors' choice of using the Multilayer Perceptron as a deep learning architecture to process the information encoded in the underlying network representation, is sub-optimal (especially for tabular data problems). In addition to this, `HNN`s impose the chordality of the underlying network and the building process of the deep neural network architecture implies the usage of non-native components inducing a substantial computational overhead while limiting its applicability to simple classification and regression problems. In this research work, we propose an alternative computational architecture that aims to solve these issues and we name it "Homological Convolutional Neural Network" (`HCNN`).

Given the adjacency matrix $\boldsymbol{A}$ constructed using IFNs (see Section 2.2), we isolate 3 different simplicial families: (i) maximal cliques with size 4 (i.e. 3-dimensional simplices or tetrahedra); (ii) maximal cliques with size 3 (i.e. 2-dimensional simplices or triangles); and (iii) maximal cliques with size 2 (i.e. 1-dimensional simplices or edges). When using the TMFG as a network representation, these 3 structures are sufficient to capture all the higher-order dependency structures characterizing the underlying system. The input of the novel deep learning architecture is hence represented by 3 different 1D vectors that we call H (i.e. realizations of the input features belonging to at least one tetrahedron), R (i.e. realizations of the input features belonging to at least one triangle), and E (i.e. realizations of the input features belonging to at least one edge) respectively. As a first step, in `HCNN`, we perform a 1D convolution across each set of features defining a realization of a simplicial family. We use a kernel size and a stride equals to $k+1$ (i.e. the dimension of the simplicial structure itself), and a number of filters $\zeta \in [4, 8, 12, 16]$ (notice that, as described in Section 3, the punctual value is chosen through an extended hyper-parameters search). This means that, given the three input vectors H, R and E representing the three simplicial families characterizing a TMFG, we compute a 1D convolution with a kernel size and a stride equal to 2, 3 and 4 respectively for edges, triangles, and tetrahedra. The usage of stride is necessary to prevent the "parameter sharing" (Zhang et al., 2019b). While generally considered an attractive property as fewer parameters are estimated and overfitting is avoided, in our case it leads to inconsistencies. Indeed, geometrical structures belonging to the same simplicial family (i.e. edges, triangles, and tetrahedra), but independent in the hierarchical dependency structure of the system, would share parameters.

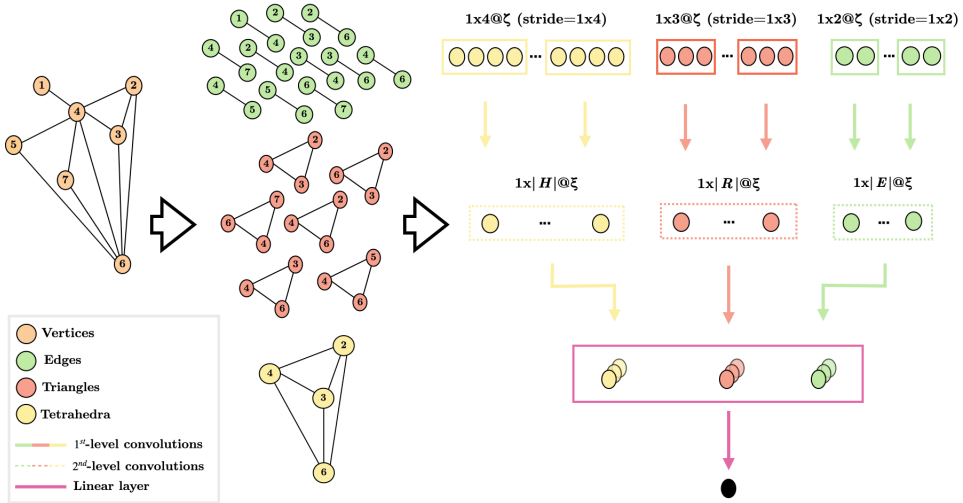


Figure 1: Pictorial representation of an HCNN and its building pipeline. From left to right, (i) we start from a chordal graph representing the dependency structures of features in the underlying system; (ii) we isolate the maximal cliques corresponding to 1-, 2- and 3-dimensional simplices (i.e. edges, triangles, tetrahedra) and we group them into 1D vectors containing features' realizations; (iii) we compute a 1st-level convolution to extract simplicial-wise non-linear relationships; (iv) we compute a 2nd-level convolution, which operates on the output of the previous level of convolution across all the representatives of each simplicial family extracting a class of non-trivial homological insights; (v) we finally apply a linear map from the 2nd-level convolution to the output, extracting model's predictions.

After the 1st-level convolution, which extract element-wise information from geometrical structures belonging to the same simplicial family, we apply a 2nd-level convolution to extract further homological insights. Indeed, the convolution is applied to the output of the first layer, extracting information related to entities belonging to the same simplicial family, which are not necessarily related in the original network representation. In this case, we use a kernel with a size equal to the cardinality of the simplicial family (i.e. $|E|$, $|R|$, $|H|$ respectively) and a number of filters $\xi \in [32 - 64]$ (notice that, also in this case, the punctual value is chosen through an extended hyper-parameters search). The final layer of the `HCNN` architecture is linear, and maps the outputs from the 2nd-level convolution to the output. It is worth noticing that each level of convolution is followed by a regularization layer with a dropout rate equal to 0.25 and the non-linear activation function is the standard Rectified Linear Unit (ReLU). As one can see, the `HCNN`'s building pipeline is fully explainable: (i) we start from a network representation (i.e. the IFN) that captures the system's multivariate probability distribution by maximising its likelihood (see Section 2.4) and gaining relational information from sparse tabular inputs; and (ii) we transform such a representation into a standalone neural network architecture by mapping each topological prior into a computational block. The only hyper-parameter subject to tuning is the number of filters in both layers of the `HCNN` (i.e. ζ and ξ); all the other architectural choices are

deterministic and led by the topological structure of the IFN describing the underlying system.

2.4. On the learning process of network's representation

According to the setup described in Section 2.3, the architecture proposed in the current paper arises from a graph-based higher-order representation $\mathcal{G}$ of a multivariate system $\mathbf{X} = (X_1, \ldots, X_n)^\top$, where the components X_i are unidimensional scalar random variables characterized by an (unknown) probability density function $f(\mathbf{X})$. Our primary goal is hence estimating the multivariate probability density function with representation structure $\mathcal{G}^*$, $\widetilde{f}(\mathbf{X}|\mathcal{G}^*)$, that best describes the true and unknown $f(\mathbf{X})$. This problem is known to be NP-hard (Osman et al., 2003), however one can restrict the search space and identify a priori the optimal network representation by analysing the dependency structure of the features characterizing the system under analysis. From an information theoretic perspective, the learning of an optimal network representation $\mathcal{G}^*$ consists of minimising the Kullback-Leibler divergence (D_{KL}) (Kullback and Leibler, 1951) between $f(\mathbf{X})$ and $\widetilde{f}(\mathbf{X}|\mathcal{G})$, and, consequently, the cross-entropy (H) of the underlying system:

$$\begin{aligned}
\mathcal{G}^* &\Rightarrow \arg\min_{\mathcal{G}} D_{KL}(f(\mathbf{X}) \mid\mid \widetilde{f}(\mathbf{X}|\mathcal{G})) \\
&\Rightarrow \arg\min_{\mathcal{G}} \cancel{\mathbb{E}_f(\log f(\mathbf{X}))} - \mathbb{E}_f(\log \widetilde{f}(\mathbf{X}|\mathcal{G})) \\
&\Rightarrow \arg\min_{\mathcal{G}} (H(\mathbf{X}|\mathcal{G}))
\end{aligned} \tag{1}$$

The term $\mathbb{E}_f(\log f(\mathbf{X}))$ in Equation (1) is independent of $\mathcal{G}$ and therefore its value is irrelevant to the purpose of discovering the optimal representation network. The second term, $-\mathbb{E}(\log \widetilde{f}(\mathbf{X}|\mathcal{G}))$ (notice the minus), instead depends on $\mathcal{G}$ and must be minimized. It is the estimate of the entropy of the multivariate system under analysis and corresponds to the so-called cross-entropy. Given that the true underlying distribution is unknown, the expectation cannot be computed exactly, however, it can be estimated with arbitrary precision using the sample mean. Such a sample mean approximates the expected value of the negative log-likelihood of the model $\widetilde{f}(\mathbf{X}|\mathcal{G})$. Therefore, the previously described optimization problem, becomes a likelihood ($\mathcal{L}$) maximization problem. The network associated with the largest system's likelihood can be constructed step-by-step by joining disconnected parts that share the largest mutual information. Indeed, in a graph, the gain achieved by joining two variables X_a and X_b, is approximately given by the mutual information shared by the two variables $\simeq I(X_a; X_b)$. In turn, at the second-order approximation, the mutual information corresponds to the squared correlation coefficient between the two variables. It follows that the gain in likelihood is $I(X_a; X_b) \simeq \rho^2_{a,b}$ (Ihara, 1993), and the TMFG construction with ρ^2 weights implies a graph that maximizes the system's likelihood itself.

3. Experiments

In this section, we compare the performance of the `HCNN` classifier in its `MeanSimMatrix` and `BootstrapNet` configuration (see Section 2.2) against 5 machine learning and 3 deep learning SOTA classifiers under homogeneous evaluation conditions. We consider `LogisticRegression`,

`RandomForest`, `XGBoost`, `LightGBM` and `CatBoost` as representatives of machine learning classifiers, and `MLP`, `TabNet` and `TabPFN` as representatives of deep learning classifiers. For each of them, the inference process is structured into two different phases: (i) the hyper-parameters search stage; and (ii) the training/test stage with optimal hyper-parameters. Both stages are repeated 10 times with fixed seeds (see Appendix A) to guarantee a full reproducibility of results. For each run, we allow for a maximum of 500 hyper-parameters search iterations allocating 8 CPUs with 2GB memory each and a time budget of 48 hours. Obtained results are statistically validated using the Wilcoxon significance test, a standard metric for comparing classifiers across multiple datasets (Demšar, 2006). On a second stage of the analysis, we investigate the scalability of each model in tackling extensive numerical tabular classification tasks. In so doing, we use an ad-hoc suite of datasets (see Section 2.1), while maintaining the inference process described earlier in this Section. A model converges (i.e. it is able to scale to larger classification problems) once completing the learning task using the given computational resources in the allocated time budget for all the 10 seeds.

3.1. Small tabular classification problems

Table 3.1 reports a cross-datasets, out-of-sample comparison of classifiers previously listed in this section. For each model, we provide (i) the average; (ii) the best/worst ranking position considering three different evaluation metrics; (iii) the average value for each evaluation metric; (iv) the time required for the hyper-parameters tuning; and (v) for the training/test run with optimal hyper-parameters. An extended version of these results is provided in Appendix E and in Appendix F.

Table 1: Cross-datasets, out-of-sample comparison of classifiers' performance. For each model, we provide (i) the average ("M." abbreviation); (ii) the best/worst ("B/W" abbreviation) ranking position considering three different evaluation metrics; (iii) the average ("M." abbreviation) value for each evaluation metric; and (iv) the time in seconds (s) required for the hyper-parameters tuning and for the training/test run with optimal hyper-parameters.

	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
M. rank F1_Score	5.333	6.333	5.500	5.277	4.666	9.500	7.388	2.388	4.888	3.722
M. rank Accuracy	4.888	5.972	6.194	5.916	5.388	9.666	7.166	1.833	4.694	3.277
M. rank MCC	5.166	6.388	5.611	5.666	4.833	9.500	7.333	2.166	4.777	3.5556
B/W rank F1_Score	1-9	1-10	1-9	1-8	1-8	8-10	2-10	1-10	1-9	2-7
B/W rank Accuracy	1-10	1-10	3-9	2-8	1-10	8-10	2-10	1-5	1-9	1-6
B/W rank MCC	1-10	2-10	1-9	1-8	1-8	8-10	2-10	1-8	1-9	2-6
M. F1_Score	0.79±0.14	0.77±0.15	0.81±0.13	0.80±0.12	0.80±0.12	0.74±0.15	0.77±0.15	0.83±0.14	0.80±0.14	0.81±0.13
M. Accuracy	0.86±0.09	0.86±0.09	0.87±0.10	0.86±0.09	0.86±0.10	0.83±0.09	0.86±0.09	0.88±0.09	0.87±0.09	0.88±0.09
M. MCC	0.68±0.24	0.65±0.27	0.69±0.24	0.68±0.23	0.68±0.24	0.59±0.29	0.64±0.28	0.71±0.26	0.68±0.26	0.70±0.25
M. Time (s) (Tune)	22.95	3097.83	842.08	900.24	3704.13	136.47	37148.33	11475.86	6349.97	7103.76
M. Time (s) (Train + Test)	0.03	1.24	0.88	4.49	17.73	0.17	35.82	15.51	17.09	15.58

From the table above, one can observe that, on average, the `TabPFN` model occupies a ranking position higher than the one of `HCNN` both in its `MeanSimMatrix` and in `BootstrapNet` configuration. However, it is worth noticing that when we evaluate models' performance through the F1_Score and the Matthew's Correlation Coefficient (MCC) (i.e. the two performance metrics that are less prone to bias induced by unbalanced datasets), the `HCNN` in the `MeanSimMatrix` configuration occupies a ranking position for the worst performance which is better than the one of its immediate competitor (i.e. position 7 and 6 for `HCNN MeanSimMatrix` vs position 10 and 8 for `TabPFN`). The same happens in the case

of the `HCNN BootstrapNet` with the F1_score. These findings highlight an evident robustness of the `HCNN` model, which is superior not only to `TabPFN` model, but also to all the other deep learning and machine learning alternatives. More generally, both `TabPFN` and `HCNN` show superior performances compared to the other two deep learning models (i.e. `MLP` and `TabNet`), which occupy an average ranking position equal to ~ 7 and ~ 9 respectively for all the three different evaluation metrics. Among machine learning models, `CatBoost` achieves the highest performance with an average ranking position equal to ~ 4 considering the F1_Score and the MCC, and equal to ~ 5 considering the accuracy score (in this case the position number 4 is occupied by `LogisticRegression`).

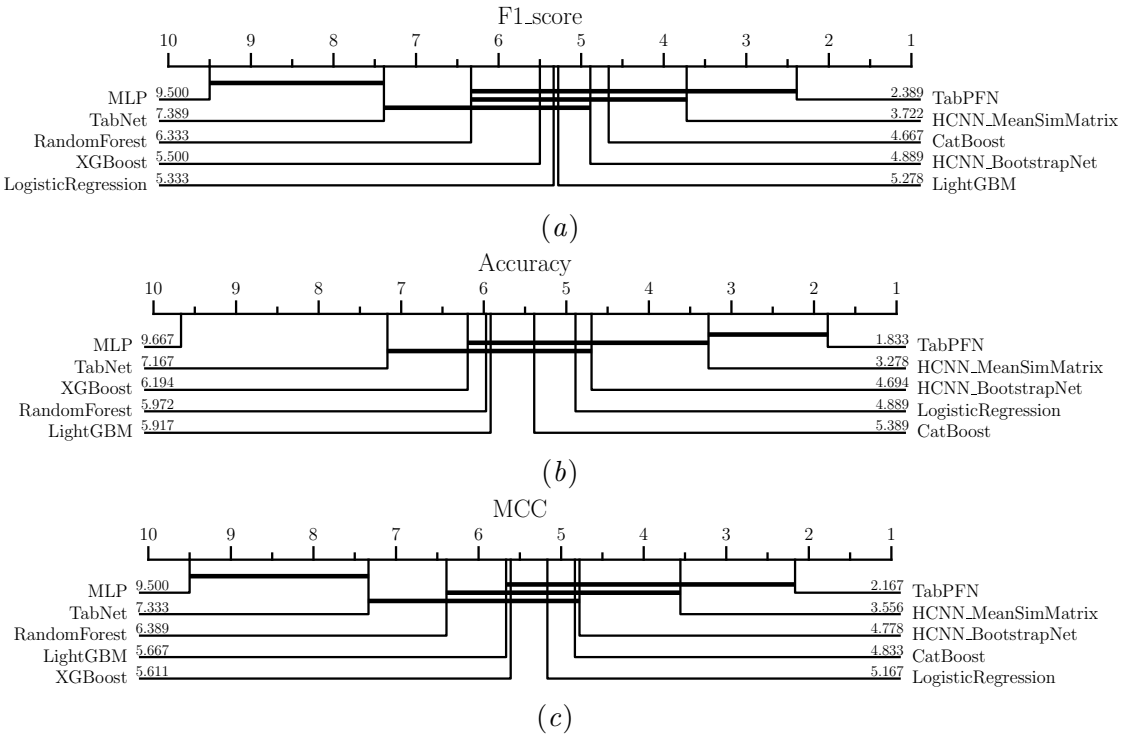


Figure 2: Critical Difference plots on out-of-sample average ranks with a Wilcoxon significance analysis. In (a) the test is run considering the F1_Score; in (b) the test is run considering the accuracy; in (c) the test is run considering the MCC.

Models' numerical performances for each evaluation metric (see Appendix F) enforce all the findings discussed above. It is, however, clear that the differences in performance are very small. To assess their statistical significance, we use the Critical Difference (CD) diagram of the ranks based on the Wilcoxon significance test (with p-values below 0.1), a standard metric for comparing classifiers across multiple datasets (Demšar, 2006). The overall empirical comparison of the methods is given in Figure 2. We notice that the performance of `HCNN` and `TabPFN` is not statistically different. This finding is coherent across the three evaluation metrics and is particularly relevant because it makes these deep learning architectures the only two which are consistently comparable with the SOTA machine learning ones. These findings legitimate the methodology proposed in the current research work as being itself a SOTA, both in terms of performance and in terms of computational com-

plexity (see Appendix G for an extended study on the number of parameters). We cannot assert the same for the `TabPFN`, which is among the SOTA models in terms of performance but it is the worst model in terms of computational (and architectural) complexity.

3.2. Models' scalability to larger tabular numerical classification problems

All the models considered in the current research work are primarily designed to handle small tabular classification problems. According to (Hollmann et al., 2022), a dataset is defined as "small" if it contains up to 2 000 samples and 100 features. However, in this section, we explore the ability of the models to scale to larger problems. In doing so, we use benchmark datasets characterized, in turn, by a number of samples greater than 2 000 or a number of features greater than 100.

Table 2: Study on models' ability to scale to larger problems. Considered datasets belong to the OpenML benchmark suite "Tabular benchmark numerical classification" (Grinsztajn et al., 2022). For each of them, we report the OpenML ID, the number of samples, and the number of features. We indicate the success in solving the corresponding tabular classification task with a (✓) symbol, while a failure to solve the problem is denoted by an (✗) symbol.

OpenML ID	# Samples	# Features	Model									
			LogisticRegression	RandomForest	XGBoost	LightGBM	CatBoost	MLP	TabNet	TabPFN	HCNN BootstrapNet	HCNN MeanSimMatrix
361055	16714	10	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361062	10082	26	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361063	13488	16	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361065	13376	10	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361066	10578	7	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361275	13272	20	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361276	3434	419	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
361277	20634	8	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
361278	10000	22	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓

From Table 2, one can observe that the proposed datasets are sufficient in underlining the issues of two models, namely the `TabPFN` model and the `HCNN` model in its `MeanSimMatrix` configuration. In the first case, the model is entirely unable to scale to problems with a larger number of samples and features. This limitation was already pointed out in the original work by (Hollmann et al., 2022) and directly depends on the model's architecture, which strongly leverage the power of attention-based mechanism. Indeed, the runtime and memory usage of the `TabPFN` architecture scales quadratically with the number of inputs (i.e. training samples passed) and the fitted model cannot work with datasets with a number of features > 100. In the case of `HCNN MeanSimMatrix`, instead, the proposed architecture demonstrates a limit in handling problems characterised by a large number of features (but not samples). Also in this case, the reason of the failure should be searched in the model's architectural design choices. Indeed, this architecture is characterised by a strong linear relationship between the number of features and the number of parameters (see Appendix G), meaning that when the first parameter is large, convolving across all representatives of each simplicial complex family becomes computationally demanding. A solution to this problem can be found in employing the `BootstrapNet` configuration, which disrupts the linear relationship discussed earlier, resulting in a significant reduction in the number of parameters when dealing with a large number of features. While this approach demonstrates considerable efficacy, it remains reliant on a threshold parameter (see Section 2.2), suggesting the need for more advanced

and parameter-free alternatives. For the seek of completeness, in Appendix H we partially repeat the analyses presented in Section 3.1 on the newly introduced datasets. Because of the fragmentation caused by the increased size, we report only the dataset-dependent analyses, excluding cross-datasets ones.

4. Conclusion

In this paper, we introduce the Homological Convolutional Neural Network (`HCNN`), a novel deep learning architecture that revisits the simpler Homological Neural Network (`HNN`) to gain abstraction, representation power, robustness, and scalability. The proposed architecture is data-centric and arises from a graph-based higher-order representation of dependency structures among multivariate input features. Compared to `HNN`, our model demonstrates a higher level of abstraction. Indeed, we have higher flexibility in choosing the initial network's representation, as we can choose from the universe simplicial complexes and we are not restricted to specific sub-families. Looking at geometrical structures at different granularity levels, we propose a clear-cut way to leverage the power of convolution on sparse data representations. This allows to fully absorb the representation power of `HNN` in the very first layer of `HCNN`, leaving room for additional data transformations at deeper levels of the architecture. Specifically, in the current research work, we build the `HCNN` using a class of information filtering networks (i.e. the TMFG) that uses squared correlation coefficients to maximize the likelihood of the underlying system. We propose two alternative architectural solutions: (i) the `MeanSimMatrix` configuration; and (ii) the `BootstrapNet` configuration. Both of them leverage the power of bootstrapping to gain robustness toward data noise and the intrinsic complexity of interactions among the underlying system's variables. We test these two modeling solutions on a set of tabular numerical classification problems (i.e. one of the most challenging tasks for deep learning models and the one where `HNN` demonstrates the poorest performances). We compare `HCNN` with different machine- and deep learning architectures, always teeing SOTA performances and demonstrating superior robustness to data unbalances. Specifically, we demonstrate that `HCNN` is able to compete with the latest transformer-based architectures (e.g. `TabPFN`) by using a considerably lower and easily controllable number of parameters (especially in the `BootstrapNet` configuration), guaranteeing a higher level of explicability in the neural network's building process and having a comparable running time without the need for pre-training. We finally propose a study on models' scalability to large datasets. We underline the fragility of transformer-based models and we demonstrate that `HCNN` in its `MeanSimMatrix` configuration is unable to manage datasets characterized by a large number of input features. In contrast, we show that the design choices adopted for the `BootstrapNet` configuration offers a parametric solution to the problem. Despite significant advances introduced by `HCNN`s, this class of neural networks remains in an embryonic phase. Further studies on underlying network representations should propose alternative similarity metrics that replace the squared correlation coefficients for mixed data-types (i.e. categorical and numerical or categorical only data-types), and further work is finally required to better understand low-level interactions captured by the proposed neural network model. This final point would certainly lead to a class of non-parametric, parsimonious `HCNN`s.

References

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

Tomaso Aste. Topological regularization with information filtering networks. *Information Sciences*, 608:655–669, 2022.

Tomaso Aste, Tiziana Di Matteo, and ST Hyde. Complex networks on hyperbolic surfaces. *Physica A: Statistical Mechanics and its Applications*, 346(1-2):20–26, 2005.

Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, Khoa Doan, and Sathiya S Keerthi. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.

Baosenguo. baosenguo/kaggle-moa-2nd-place-solution. https://github.com/baosenguo/Kaggle-MoA-2nd-Place-Solution, 2021. Online; accessed 07-June-2023.

Wolfram Barfuss, Guido Previde Massara, Tiziana Di Matteo, and Tomaso Aste. Parsimonious modeling with information filtering networks. *Physical Review E*, 94(6):062306, 2016.

Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54, 2018.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml: A benchmarking layer on top of openml to quickly create, download, and share systematic benchmarks. *NeurIPS*, 2021. URL https://openreview.net/forum?id=OCrD8ycKjG.

Ankita Bose and BK Tripathy. Deep learning for audio signal classification. *Deep learning research and applications*, pages 105–136, 2020.

Antonio Briola and Tomaso Aste. Dependency structures in cryptocurrency market from high to low frequency. *arXiv preprint arXiv:2206.03386*, 2022.

Antonio Briola and Tomaso Aste. Topological feature selection. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 534–556. PMLR, 2023.

Antonio Briola, David Vidal-Tomás, Yuanrong Wang, and Tomaso Aste. Anatomy of a stablecoin's failure: The terra-luna case. *Finance Research Letters*, page 103358, 2022.

Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2): 1153–1176, 2015.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

Dheeru Dua, Casey Graff, et al. Uci machine learning repository. 2017.

Bradley Efron, Elizabeth Halloran, and Susan Holmes. Bootstrap confidence levels for phylogenetic trees. *Proceedings of the National Academy of Sciences*, 93(23):13429–13429, 1996.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Shunsuke Ihara. *Information theory for continuous systems*, volume 2. World Scientific, 1993.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. *arXiv preprint arXiv:2106.11189*, 536, 2021a.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34: 23928–23941, 2021b.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Rosario N Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

Guido Previde Massara, Tiziana Di Matteo, and Tomaso Aste. Network filtering for big data: Triangulated maximally filtered graph. *Journal of complex Networks*, 5(2):161–178, 2017.

Jean Jacques Ohana, Steve Ohana, Eric Benhamou, David Saltiel, and Beatrice Guez. Explainable ai (xai) models applied to the multi-agent environment of financial markets. In *Explainable and Transparent AI and Multi-Agent Systems: Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers 3*, pages 189–207. Springer, 2021.

Ibrahim H Osman, Baydaa Al-Ayoubi, and Musbah Barake. A greedy random adaptive search procedure for the weighted maximal planar graph problem. *Computers & industrial engineering*, 45(4):635–651, 2003.

Myeongsuk Pak and Sanghoon Kim. A review of deep learning in image recognition. In *2017 4th international conference on computer applications and information processing technology (CAIPT)*, pages 1–3. IEEE, 2017.

Guansong Pang, Charu Aggarwal, Chunhua Shen, and Nicu Sebe. Editorial deep learning for anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2282–2286, 2022.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.

Danda B Rawat, Ronald Doku, and Moses Garuba. Cybersecurity in big data era: From securing big data to data-driven security. *IEEE Transactions on Services Computing*, 14 (6):2055–2072, 2019.

Swati Sachan, Jian-Bo Yang, Dong-Ling Xu, David Eraso Benavides, and Yang Li. An explainable ai decision-support-system to automate loan underwriting. *Expert Systems with Applications*, 144:113100, 2020.

Vsevolod Salnikov, Daniele Cassese, and Renaud Lambiotte. Simplicial complexes and complex systems. *European Journal of Physics*, 40(1):014001, 2018.

Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *Advances in Neural Information Processing Systems*, 31, 2018.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Ravid Shwartz-Ziv, Amichai Painsky, and Naftali Tishby. Representation compression and generalization in deep neural networks, 2018.

Sulaiman Somani, Adam J Russak, Felix Richter, Shan Zhao, Akhil Vaid, Fayzan Chaudhry, Jessica K De Freitas, Nidhi Naik, Riccardo Miotto, Girish N Nadkarni, et al. Deep learning and the electrocardiogram: review of the current state-of-the-art. *EP Europace*, 23(8):1179–1191, 2021.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.

Joaquín J Torres and Ginestra Bianconi. Simplicial complexes: higher-order spectral dimension and dynamics. *Journal of Physics: Complexity*, 1(1):015002, 2020.

Michele Tumminello, Tomaso Aste, Tiziana Di Matteo, and Rosario N Mantegna. A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences*, 102(30):10421–10426, 2005.

Michele Tumminello, Claudia Coronnello, Fabrizio Lillo, Salvatore Micciche, and Rosario N Mantegna. Spanning trees and bootstrap reliability estimation in correlation-based networks. *International Journal of Bifurcation and Chaos*, 17(07):2319–2329, 2007.

Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. Trust issues: Uncertainty estimation does not enable reliable ood detection on medical tabular data. In *Machine Learning for Health*, pages 341–354. PMLR, 2020.

David Vidal-Tomás, Antonio Briola, and Tomaso Aste. Ftx's downfall and binance's consolidation: the fragility of centralized digital finance. *arXiv preprint arXiv:2302.11371*, 2023.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

Siqi Wang, Jiyuan Liu, Guang Yu, Xinwang Liu, Sihang Zhou, En Zhu, Yuexiang Yang, Jianping Yin, and Wenjing Yang. Multiview deep anomaly detection: A systematic exploration. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Yuanrong Wang, Antonio Briola, and Tomaso Aste. Homological neural networks: A sparse architecture for multivariate complexity, 2023a.

Yuanrong Wang, Antonio Briola, and Tomaso Aste. Topological portfolio selection and optimization. *arXiv preprint arXiv:2310.14881*, 2023b.

Min Zhang and Juntao Li. A commentary of gpt-3 in mit technology review 2021. *Fundamental Research*, 1(6):831–833, 2021.

Qi Zhang, Longbing Cao, Chongyang Shi, and Zhendong Niu. Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5125–5137, 2021.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019a.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019b.

Appendix A.

Tables 3 and 4 report an overview of the main characteristics of the two suites of benchmark datasets used in the current research work. In both cases the open-access of data is guaranteed by OpenML (Bischl et al., 2021). Training/validation/test split is not provided. For all the datasets, the 50% of the raw dataset is used as a training set, the 25% as validation set, and the remaining 25% as a test set. To prove the statistical significance of results presented in the current research work, all the analyses are repeated on 10 different combinations of training/validation/test splits. The reproducibility of results is guaranteed by a rigorous usage of seeds (i.e. [12, 190, 903, 7 687, 8 279, 9 433, 12 555, 22 443, 67 822, 9 822 127]).

Table 3: Datasets used for models' evaluation. These include 18 numerical tabular datasets from the OpenML-CC18 benchmark suite with at most 2 000 samples, 100 features and 10 classes. For each dataset we report the name, the number of input features, the number of samples before training/validation/test split, the number of classes, the number of samples by class and, finally, the corresponding ID number in the OpenML benchmark suite.

Dataset Name	**# Features**	**# Samples**	**# Classes**	**Samples by Class**	**OpenML ID**
balance-scale	4	625	3	49/288/288	11
mfeat-fourier	76	2,000	10	200/200/200/200/200/ 200/200/200/200/200	14
breast-w	9	683	2	444/239	15
mfeat-karhunen	64	2,000	10	200/200/200/200/200/ 200/200/200/200/200	16
mfeat-morphological	6	2,000	10	200/200/200/200/200/ 200/200/200/200/200	18
mfeat-zernike	47	2,000	10	200/200/200/200/200/ 200/200/200/200/200	22
diabetes	8	768	2	500/268	37
vehicle	18	846	4	218/212/217/199	54
analcatdata_authorship	70	841	4	317/296/55/173	458
pc4	37	1458	2	1280/178	1049
kc2	21	522	2	415/107	1063
pc1	21	1109	2	1032/77	1068
banknote-authentication	4	1372	2	762/610	1462
blood-transfusion-service-center	4	748	2	570/178	1464
qsar-biodeg	41	1055	2	699/356	1494
wdbc	30	569	2	357/212	1510
steel-plates-fault	27	1941	7	402/55/391/ 673/158/72/190	40982
climate-model-simulation-crashes	18	540	2	46/494	40994

The first set of data belongs to the "OpenML-CC18" benchmark suite (Hollmann et al., 2022) and is used to compare models' performance in solving numerical classification prob-

lems, while the second set of data belongs to the "OpenML tabular benchmark numerical classification" suite (Grinsztajn et al., 2022) and is used to test the models' ability to scale to larger problems of the same type. Looking at Table 3, we notice that the average number of features is 28.5 with a standard deviation equal to 22.2. The dataset with the lowest number of features is "balance-scale" (i.e. 4 features) with OpenML ID 11. The dataset with the largest number of features is "mfeat-fourier" (i.e. 76 features) with OpenML ID 14. All the features in all the datasets are numerical only and no missing values are detected. The average number of samples is 1191.6 with a standard deviation equal to 557.3. The dataset with the lowest number of samples is "kc2" (i.e. 522 samples) with OpenML ID 1063. The dataset with the largest number of samples is "mfeat-morphological" (i.e. 2 000 samples) with OpenML ID 18. 11 of the considered datasets are binary, while 8 are multi-class.

Table 4: Datasets used to test models' scalability. These include 9 numerical tabular datasets from the OpenML "Tabular benchmark numerical classification" suite (Grinsztajn et al., 2022). All these datasets violate at least one of the selection criteria in (Hollmann et al., 2022) (i.e. they are characterised by a number of samples $> 2\,000$ or they are characterised by a number of features > 100). For each dataset we report the name, the number of input features, the number of samples before training/validation/test split, the number of classes, the number of samples by class and, finally, the corresponding ID number in the OpenML benchmark suite.

Dataset Name	**# Features**	**# Samples**	**# Classes**	**Samples by Class**	**OpenML ID**
credit	10	16714	2	8357/8357	361055
pol	26	10082	2	5041/5041	361062
house_16H	16	13488	2	6744/6744	361063
MagicTelescope	10	13376	2	6688/6688	361065
bank-marketing	7	10578	2	5289/5289	361066
default-of-credit-card-clients	20	13272	2	6636/6636	361275
Bioresponse	419	3434	2	1717/1717	361276
california	8	20634	2	10317/10317	361277
heloc	22	10000	2	5000/5000	361278

Looking at Table 4, we notice that the average number of features is 59.8 with a standard deviation equal to 127.2. The dataset with the lowest number of features is "bank-marketing" (i.e. 7 features) with OpenML ID 361066. The dataset with the largest number of features is "Bioresponse" (i.e. 419 features) with OpenML ID 361276. All the features in all the datasets are numerical only and no missing values are detected. The average number of samples is 12397.6 with a standard deviation equal to 4523.4. The dataset with the lowest number of samples is "Bioresponse" (i.e. 3434 samples) with OpenML ID 361276. The dataset with the largest number of samples is "california" (i.e. 20634 samples) with OpenML ID 361277. All the considered datasets are binary.

Appendix B.

Algorithm 1: TMFG built on the similarity matrix $\hat{\mathbf{C}}$ to maximize global properties of the system under analysis (e.g., likelihood).

Data: Similarity matrix $\hat{\mathbf{C}} \in \mathbb{R}^{n,n}$ from a set of observations $\{x_{1,1}, \ldots, x_{s,1}\}, \{x_{1,2}, \ldots, x_{s,2}\} \ldots \{x_{1,n}, \ldots, x_{s,n}\}$.
Result: Sparse adjacency matrix $\boldsymbol{A}$ describing the TMFG.
Function *MaximumGain($\hat{\mathbf{C}}$, $\mathcal{V}$, t)*:
  Initialize a vector of zeros $g \in \mathbb{R}^{1\times n}$
  for $j \in t$ **do**
    for $v \notin \mathcal{V}$ **do**
      $\hat{\mathbf{C}}_{v,j} = 0$
    end
    $g = g \oplus \hat{\mathbf{C}}_{v,j}$
  end
  return $\max\{g\}$
Initialize four empty sets: $\mathcal{C}$ (cliques), $\mathcal{T}$ (triangles), $\mathcal{S}$ (separators), and $\mathcal{V}$ (vertices)
Initialize an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n,n}$ with all zeros
$\mathcal{C}_1 \leftarrow$ tetrahedron, $\{v_1, v_2, v_3, v_4\}$, obtained by choosing the 4 entries of $\hat{\mathbf{C}}$ maximizing the similarity among features
$\mathcal{T} \leftarrow$ the four triangular faces in $\mathcal{C}_1$: $\{v_1, v_2, v_3\}$, $\{v_1, v_2, v_4\}$, $\{v_1, v_3, v_4\}$, $\{v_2, v_3, v_4\}$
$\mathcal{V} \leftarrow$ Assign to $\mathcal{V}$ the remaining $n-4$ vertices not in $\mathcal{C}_1$
while $\mathcal{V}$ *is not empty* **do**
  Find the combination of $\{v_a, v_b, v_c\} \in \mathcal{T}$ (i.e. t) and $v_d \in \mathcal{V}$ which maximizes `MaximumGain(Ĉ, 𝒱, t)`
  /* $\{v_a, v_b, v_c, v_d\}$ is a new 4-clique $\mathcal{C}$, $\{v_a, v_b, v_c\}$ becomes a separator $\mathcal{S}$, three new triangular faces, $\{v_a, v_b, v_d\}$, $\{v_a, v_c, v_d\}$, and $\{v_b, v_c, v_d\}$ are created */
  Remove v_d from $\mathcal{V}$
  Remove $\{v_a, v_b, v_c\}$ from $\mathcal{T}$
  Add $\{v_a, v_b, v_d\}$, $\{v_a, v_c, v_d\}$, and $\{v_b, v_c, v_d\}$ to $\mathcal{T}$
end
for *each pair of nodes i, j in $\mathcal{C}$* **do**
  Set $\boldsymbol{A}_{i,j} = 1$
end
return $\boldsymbol{A}$

Appendix C.

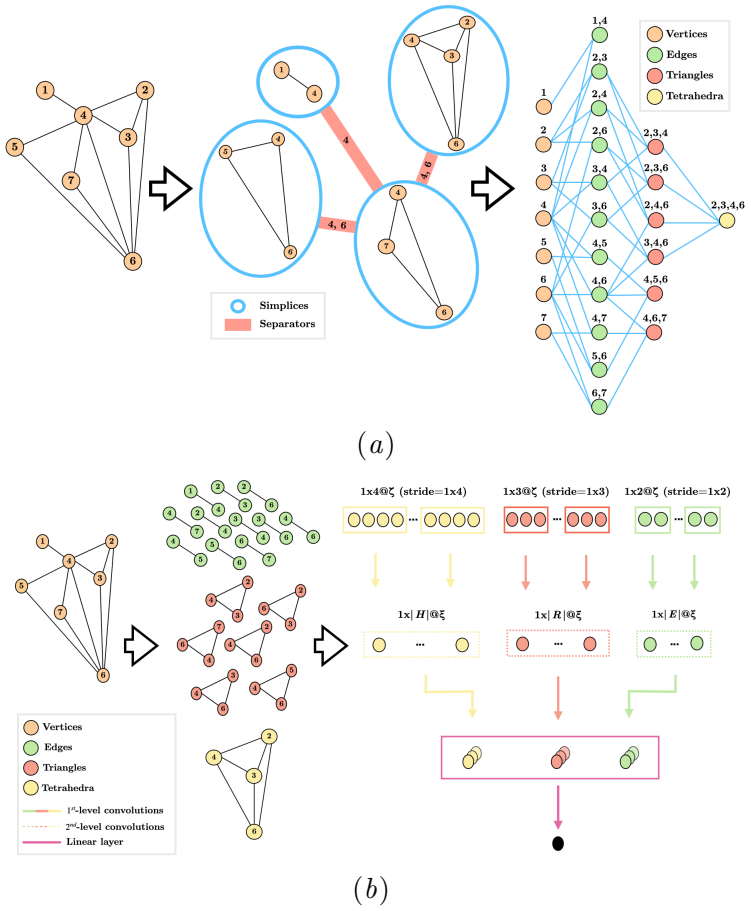


Figure 3: Pictorial representations of an HNN and HCNN architectures. In Figure (a), from left to right, (i) we start from a chordal graph representing the dependency structures of features in the underlying system; (ii) we re-arrange the network's representation to highlight the underlying simplicial complex structures (i.e. edges, triangles, tetrahedra); and (iii) we finally report a layered representation, which explicitly takes into account higher order sub-structures and their interconnections, and can be easily converted into a computational unit (i.e. a sparse MLP). In Figure (b), from left to right, (i) we start from a chordal graph representing the dependency structures of features in the underlying system; (ii) we isolate the maximal cliques corresponding to 1-, 2- and 3-dimensional simplices (i.e. edges, triangles, tetrahedra) and we group them into 1D vectors containing features' realizations; (iii) we compute a 1st-level convolution to extract simplicial-wise non-linear relationships; (iv) we compute a 2nd-level convolution, which operates on the output of the previous level of convolution across all the representatives of each simplicial family extracting a class of non-trivial homological insights; (v) we finally apply a linear map from the 2nd-level convolution to the output, extracting model's predictions.

Appendix D.

Table 5: Hyper-parameters' search space for all the classifiers considered in the current paper. When possible, search space is inherited from (Shwartz-Ziv et al., 2018) and (Hollmann et al., 2022).

Model	Name	Type	Value	Skip
LogisticRegression	penalty	cat	(l1, l2, elasticnet)	-
	max_iter	int	(100, 500, 1000)	-
RandomForest	n_estimators	int	[100, 4000]	200
	max_depth	int	[10, 50]	10
	min_samples_leaf	int	[2, 10]	2
	min_samples_split	int	[2, 10]	2
XGBoost	n_estimators	int	[100, 4000]	200
	max_depth	int	[1, 10]	3
	learning_rate	float	[e^{-4}, 1]	-
	subsample	float	[0.2, 1]	-
	colsample_bytree	float	[0.2, 1]	-
	colsample_bylevel	float	[0.2, 1]	-
	alpha	float	[e^{-4}, e^{2}]	-
	lambda	float	[e^{-4}, e^{2}]	-
	gamma	float	[e^{-4}, e^{2}]	-
CatBoost	n_estimators	int	[100, 4000]	300
	max_depth	int	[1, 10]	3
	learning_rate	float	[e^{-4}, 1]	-
	random_strength	int	[1, 10]	3
	l2_leaf_reg	int	[1, 10]	3
	bagging_temperature	float	[0, 1]	-
	leaf_estimation_iterations	int	[1, 10]	3
LightGBM	n_estimators	int	[100, 4000]	300
	max_depth	int	[1, 10]	3
	learning_rate	float	[e^{-4}, 1]	-
	num_leaves	int	[5, 50]	5
	reg_alpha	float	(0, 0.01, 1, 2, 5, 7, 10, 50, 100)	-
	reg_lambda	float	(0, 0.01, 1, 5, 10, 20, 50, 100)	-
	subsample	float	[0.2, 0.8]	-
MLP	hidded_layer_sizes	int	(10, 50, 100, 150, 200)	-
	alpha	float	(0.1, 0.01, 0.001, 0.0001)	-
	max_iter	int	(100, 500, 1000)	-
	learning_rate	cat	(constant, invscaling, adaptive)	-
TabNet	learning_rate	float	[e^{-4}, 1]	-
	n_steps	int	[1, 8]	-
	relaxation_factor	float	[0.3, 2]	-
TabPFN	n_ensemble_configurations	int	[8, 128]	8
HCNN BootstrapNet	n_filters_l1	int	[4, 16]	4
	n_filters_l2	int	[32, 64]	4
	tmfg_iterations	int	[100, 1000]	300
	tmfg_confidence	float	(0.90, 0.95, 0.99)	-
	tmfg_similarity	cat	(pearson, spearman)	-
HCNN MeanSimMatrix	n_filters_l1	int	[4, 16]	4
	n_filters_l2	int	[32, 64]	4
	tmfg_iterations	int	[100, 1000]	300
	tmfg_similarity	cat	(pearson, spearman)	-

Appendix E.

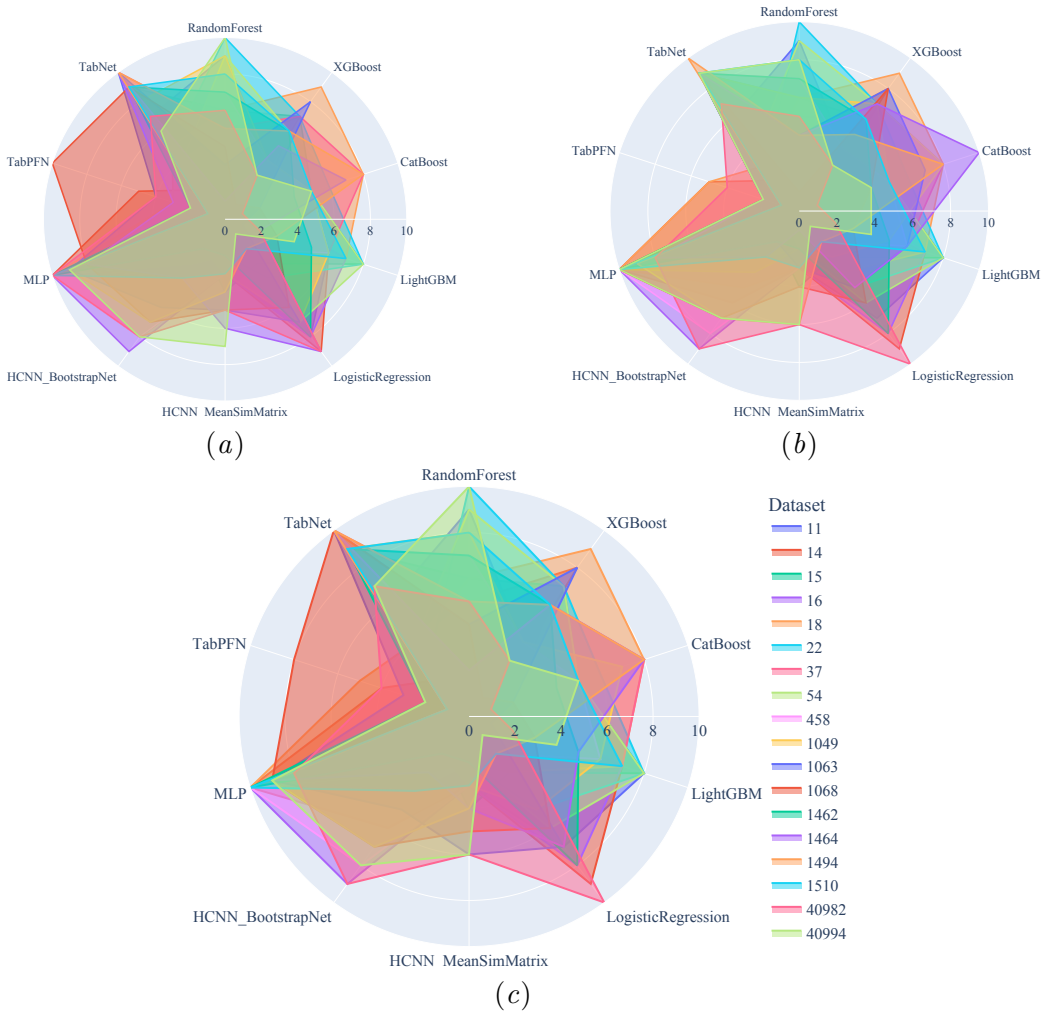


Figure 4: Out-of-sample model- and dataset-dependent average ranking considering (a) F1_Score, (b) Accuracy, and (c) MCC evaluation metric. This representation allows to clearly assess the higher robustness of HCNN model to datasets' unbalance over all its deep learning and machine learning alternatives.

All the findings related to Table 1 and discussed in Section 3.1 can be graphically visualized in Figure 4. Specifically, the highest robustness of `HCNN` model in its `MeanSimMatrix` configuration compared to `TabPFN` model can be observed in Figure 4(a) and Figure 4(c). They report the ranking position of each model on each dataset belonging to the "OpenML-CC18" benchmark suite using the F1_Score and the MCC as performance metrics respectively. In the first case, we notice that the worst ranking position by `HCNN` is 7 and is reached when dealing with dataset "climate-model-simulation-crashes" (OpenML ID 40994), while the one occupied by `TabPFN` is 10 with dataset "pc_1" (OpenML ID 1068). In the sec-

ond case, we notice that the worst performance by `HCNN` has ranking 6 and is reached when dealing with datasets "mfeat-karhunen" (OpenML ID 16), "steel-plates-fault" (OpenML ID 40982) and "climate-model-simulation-crashes" (OpenML ID 40994), while the one occupied by `TabPFN` is 8 with dataset "pc_1" (OpenML ID 1068). Except for "mfeat-karhunen" dataset (OpenML ID 16), all the datasets listed before are strongly unbalanced.

Appendix F.

Table 6: For each considered model, we report the average F1_score and the corresponding standard deviation (across 10 different seeds) on the 18 benchmark datasets from the "OpenML-CC-18" benchmark suite.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
11	0.62±0.03	0.60±0.01	0.93±0.03	0.71±0.07	0.73±0.04	0.59±0.03	0.65±0.05	0.97±0.02	0.95±0.04	0.94±0.04
14	0.79±0.01	0.82±0.02	0.81±0.02	0.81±0.02	0.82±0.01	0.77±0.03	0.82±0.01	0.82±0.01	0.81±0.01	0.82±0.02
15	0.97±0.01	0.96±0.01	0.96±0.01	0.96±0.01	0.96±0.01	0.95±0.02	0.95±0.02	0.97±0.01	0.97±0.01	0.96±0.01
16	0.95±0.01	0.96±0.01	0.95±0.00	0.95±0.01	0.96±0.01	0.94±0.01	0.96±0.01	0.97±0.01	0.94±0.01	0.96±0.01
18	0.73±0.02	0.72±0.02	0.71±0.03	0.72±0.01	0.72±0.02	0.68±0.02	0.72±0.02	0.73±0.02	0.73±0.02	0.73±0.02
22	0.82±0.01	0.77±0.02	0.79±0.01	0.78±0.01	0.79±0.02	0.77±0.03	0.80±0.03	0.82±0.02	0.82±0.02	0.82±0.01
37	0.73±0.04	0.72±0.03	0.72±0.04	0.72±0.03	0.71±0.03	0.70±0.04	0.68±0.08	0.73±0.03	0.72±0.04	0.72±0.02
54	0.76±0.03	0.73±0.03	0.77±0.04	0.76±0.04	0.77±0.04	0.67±0.06	0.80±0.04	0.84±0.03	0.82±0.03	0.83±0.03
458	1.00±0.00	0.98±0.01	0.99±0.01	0.98±0.01	0.99±0.01	0.95±0.04	0.96±0.03	1.00±0.00	0.97±0.04	0.99±0.01
1049	0.73±0.03	0.69±0.04	0.75±0.03	0.74±0.05	0.74±0.04	0.63±0.08	0.70±0.10	0.77±0.04	0.75±0.03	0.75±0.04
1063	0.68±0.05	0.70±0.04	0.64±0.11	0.71±0.05	0.70±0.03	0.62±0.09	0.58±0.09	0.69±0.06	0.68±0.06	0.68±0.05
1068	0.57±0.03	0.60±0.06	0.64±0.04	0.63±0.05	0.64±0.06	0.52±0.05	0.52±0.05	0.52±0.05	0.54±0.05	0.58±0.03
1462	0.98±0.01	0.99±0.00	0.99±0.00	1.00±0.00	1.00±0.00	0.94±0.05	0.97±0.04	1.00±0.00	1.00±0.00	1.00±0.00
1464	0.56±0.06	0.64±0.05	0.62±0.04	0.64±0.03	0.60±0.05	0.56±0.08	0.54±0.10	0.63±0.06	0.62±0.05	0.61±0.06
1494	0.85±0.02	0.83±0.02	0.83±0.02	0.84±0.02	0.82±0.02	0.81±0.04	0.78±0.13	0.86±0.01	0.83±0.02	0.84±0.02
1510	0.97±0.02	0.94±0.02	0.95±0.02	0.94±0.02	0.95±0.01	0.93±0.02	0.93±0.02	0.97±0.01	0.96±0.02	0.96±0.02
40982	0.71±0.02	0.76±0.03	0.79±0.02	0.79±0.02	0.79±0.02	0.70±0.05	0.75±0.02	0.79±0.01	0.72±0.02	0.76±0.03
40994	0.83±0.05	0.49±0.03	0.74±0.05	0.73±0.05	0.72±0.07	0.51±0.07	0.69±0.12	0.82±0.05	0.65±0.11	0.68±0.10

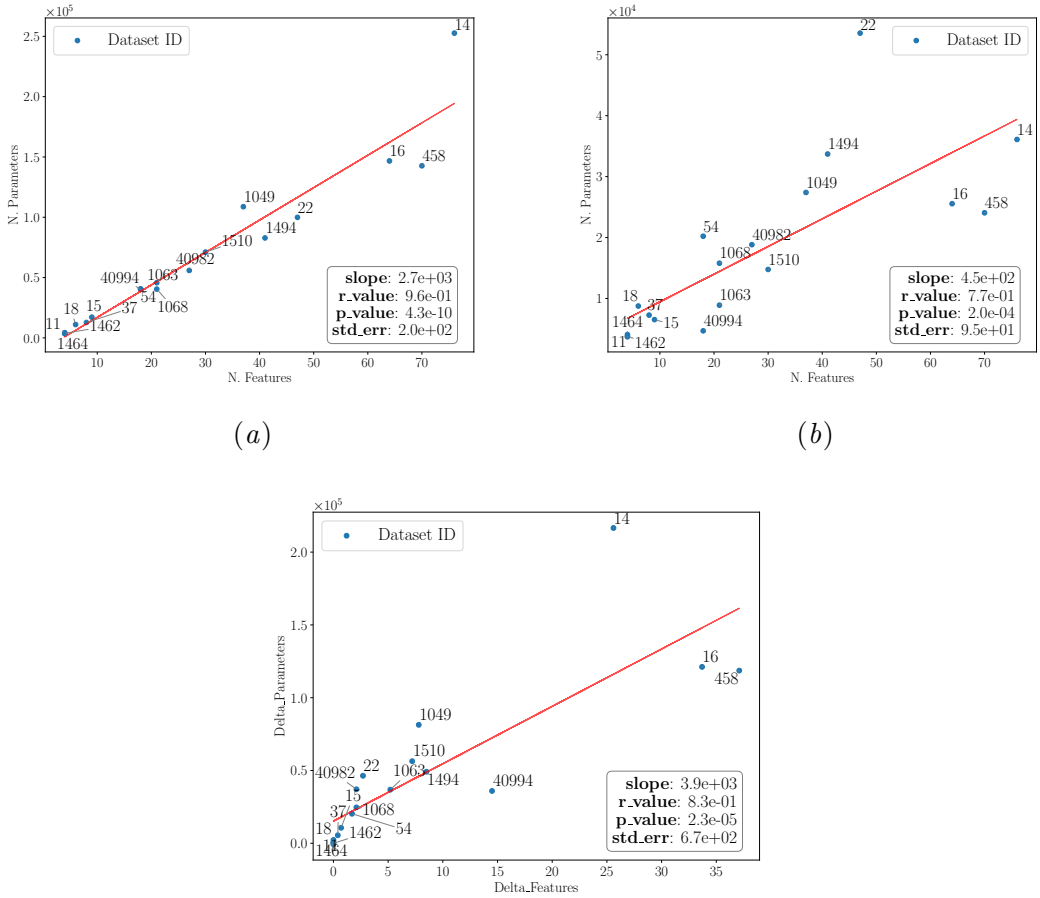
Table 7: For each considered model, we report the average Accuracy score and the corresponding standard deviation (across 10 different seeds) on the 18 benchmark datasets from the "OpenML-CC-18" benchmark suite.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
11	0.88±0.02	0.86±0.02	0.97±0.01	0.87±0.08	0.90±0.02	0.85±0.04	0.89±0.02	0.99±0.01	0.98±0.01	0.98±0.01
14	0.79±0.01	0.82±0.02	0.81±0.02	0.81±0.01	0.82±0.02	0.77±0.03	0.82±0.01	0.81±0.02	0.81±0.01	0.82±0.01
15	0.97±0.01	0.96±0.01	0.96±0.01	0.96±0.01	0.97±0.01	0.95±0.02	0.96±0.02	0.97±0.01	0.97±0.01	0.97±0.01
16	0.95±0.01	0.96±0.01	0.95±0.01	0.95±0.01	0.96±0.01	0.94±0.01	0.96±0.01	0.97±0.01	0.94±0.01	0.96±0.01
18	0.74±0.02	0.72±0.02	0.71±0.02	0.72±0.02	0.72±0.02	0.70±0.02	0.74±0.01	0.73±0.02	0.74±0.02	0.74±0.02
22	0.82±0.02	0.77±0.02	0.79±0.01	0.78±0.02	0.79±0.02	0.77±0.03	0.82±0.02	0.82±0.02	0.82±0.02	0.82±0.01
37	0.76±0.03	0.75±0.03	0.74±0.04	0.74±0.03	0.73±0.03	0.73±0.03	0.73±0.05	0.76±0.03	0.75±0.04	0.76±0.02
54	0.76±0.03	0.73±0.03	0.76±0.04	0.75±0.04	0.76±0.04	0.68±0.06	0.80±0.04	0.84±0.03	0.82±0.04	0.82±0.03
458	1.00±0.00	0.99±0.01	0.99±0.00	0.99±0.01	0.99±0.01	0.96±0.03	0.97±0.02	1.00±0.00	0.98±0.02	0.99±0.01
1049	0.90±0.01	0.90±0.02	0.90±0.01	0.90±0.02	0.89±0.02	0.88±0.01	0.89±0.02	0.91±0.01	0.90±0.01	0.90±0.01
1063	0.82±0.03	0.82±0.03	0.80±0.03	0.82±0.03	0.81±0.02	0.76±0.06	0.80±0.02	0.82±0.04	0.82±0.04	0.82±0.02
1068	0.93±0.01	0.94±0.01	0.93±0.02	0.93±0.01	0.93±0.01	0.88±0.08	0.91±0.04	0.93±0.01	0.93±0.01	0.93±0.01
1462	0.98±0.01	0.99±0.00	0.99±0.00	1.00±0.00	1.00±0.00	0.94±0.05	0.97±0.04	1.00±0.00	1.00±0.00	1.00±0.00
1464	0.78±0.03	0.79±0.02	0.77±0.04	0.78±0.02	0.74±0.03	0.76±0.05	0.76±0.05	0.79±0.03	0.79±0.02	0.79±0.02
1494	0.87±0.01	0.86±0.02	0.85±0.02	0.86±0.02	0.85±0.02	0.83±0.04	0.83±0.07	0.88±0.01	0.85±0.02	0.85±0.01
1510	0.97±0.01	0.95±0.02	0.95±0.01	0.95±0.02	0.96±0.01	0.93±0.02	0.94±0.02	0.97±0.01	0.96±0.01	0.96±0.02
40982	0.71±0.02	0.75±0.03	0.77±0.02	0.78±0.02	0.78±0.01	0.72±0.02	0.73±0.01	0.77±0.02	0.71±0.02	0.75±0.01
40994	0.95±0.01	0.91±0.02	0.94±0.01	0.93±0.01	0.93±0.02	0.84±0.11	0.91±0.02	0.95±0.01	0.93±0.03	0.93±0.02

Table 8: For each considered model, we report the average MCC and the corresponding standard deviation (across 10 different seeds) on the 18 benchmark datasets from the "OpenML-CC-18" benchmark suite.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
11	0.79±0.03	0.75±0.03	0.95±0.02	0.78±0.11	0.83±0.04	0.73±0.06	0.80±0.04	0.98±0.01	0.96±0.02	0.96±0.02
14	0.77±0.02	0.80±0.02	0.79±0.02	0.79±0.02	0.80±0.02	0.75±0.03	0.80±0.01	0.80±0.02	0.79±0.02	0.80±0.02
15	0.93±0.02	0.92±0.02	0.92±0.02	0.92±0.03	0.92±0.03	0.90±0.04	0.91±0.04	0.94±0.02	0.93±0.02	0.93±0.02
16	0.94±0.01	0.95±0.01	0.95±0.01	0.94±0.01	0.96±0.01	0.93±0.01	0.95±0.01	0.96±0.01	0.94±0.01	0.96±0.01
18	0.71±0.02	0.69±0.03	0.68±0.03	0.69±0.02	0.68±0.03	0.67±0.02	0.71±0.01	0.70±0.02	0.72±0.03	0.71±0.02
22	0.80±0.02	0.74±0.02	0.76±0.01	0.76±0.02	0.77±0.02	0.75±0.03	0.80±0.03	0.80±0.02	0.80±0.02	0.80±0.01
37	0.47±0.08	0.45±0.07	0.44±0.09	0.44±0.06	0.42±0.06	0.41±0.07	0.40±0.10	0.48±0.07	0.45±0.09	0.46±0.05
54	0.68±0.05	0.65±0.05	0.68±0.06	0.67±0.06	0.69±0.06	0.58±0.08	0.74±0.05	0.78±0.04	0.76±0.05	0.76±0.04
458	0.99±0.01	0.98±0.01	0.99±0.01	0.98±0.01	0.98±0.01	0.94±0.04	0.96±0.02	1.00±0.00	0.96±0.03	0.99±0.01
1049	0.50±0.05	0.45±0.07	0.50±0.05	0.49±0.10	0.49±0.08	0.32±0.16	0.43±0.18	0.57±0.07	0.52±0.05	0.51±0.08
1063	0.39±0.11	0.41±0.08	0.32±0.17	0.42±0.11	0.42±0.06	0.30±0.15	0.25±0.15	0.41±0.12	0.40±0.12	0.40±0.09
1068	0.20±0.05	0.28±0.12	0.30±0.08	0.30±0.10	0.30±0.13	0.07±0.10	0.06±0.09	0.09±0.13	0.14±0.13	0.25±0.08
1462	0.96±0.01	0.98±0.01	0.99±0.01	0.99±0.01	1.00±0.01	0.88±0.10	0.94±0.08	1.00±0.00	1.00±0.00	1.00±0.00
1464	0.24±0.08	0.33±0.09	0.27±0.08	0.30±0.06	0.22±0.09	0.19±0.13	0.18±0.16	0.33±0.07	0.31±0.07	0.31±0.08
1494	0.70±0.03	0.67±0.04	0.67±0.04	0.67±0.03	0.65±0.04	0.63±0.07	0.60±0.20	0.73±0.03	0.66±0.04	0.67±0.03
1510	0.93±0.03	0.88±0.04	0.90±0.03	0.89±0.04	0.90±0.02	0.86±0.04	0.87±0.04	0.93±0.02	0.92±0.03	0.92±0.03
40982	0.62±0.03	0.68±0.03	0.70±0.03	0.71±0.02	0.72±0.02	0.63±0.03	0.65±0.02	0.70±0.02	0.62±0.03	0.67±0.02
40994	0.68±0.10	0.03±0.10	0.53±0.08	0.51±0.09	0.49±0.13	0.07±0.11	0.39±0.25	0.65±0.11	0.38±0.24	0.43±0.18

Appendix G.



(c)

Figure 5: Study of the relationship between the number of total features (x-axis) and the number of parameters (y-axis) for the (a) HCNN MeanSimMatrix and (b) HCNN BootstrapNet configuration. Figure (c) reports the relationship structure among the difference in the number of features (x-axis) and the difference in the number of total parameters (y-axis) when using the two above-mentioned configurations.

Figure 5 reports the relationship between the number of features and the total number of parameters for the `HCNN` in its `MeanSimMatrix` configuration, the relationship between the number of features and the total number of parameters for the `HCNN` in its `BootstrapNet` configuration and the relationship between the difference in the number of features and the difference in the total number of parameters in the two configurations. Looking at Figure 5(a), it is possible to conclude that a strong linear relationship exists between the number of features and the total number of parameters of the `HCNN` model in the `MeanSimMatrix` configuration. This finding was expected since the proposed model's architecture entirely

depends on the complete homological structure of the underlying system. This means that, each time a new feature is introduced, we could potentially observe an increase in the number of edges, triangles, and tetrahedra, which in turn determines a proportional increase in the number of parameters of the `HCNN` itself. On this point, we need to underline that the magnitude of the slope of the regression line heavily depends on the optimal hyper-parameters describing the number of filters in the two convolutional layers. Looking at Figure 5(*b*), we observe again a relatively strong linear relationship between the number of features and the total number of parameters of the `HCNN` model in the `BootstrapNet` configuration. The difference in r_value between the two configurations is equal to 0.19 and depends on the fact that, in the second case, the optimal threshold value which maximizes the model's performance (different across datasets), does not depend on the number of inputs features and determines the ablation of a number of features that has no dependence with any other factor. More generally, in the `BootstrapNet` configuration, we observe a number of parameters that is, on average, one order of magnitude below the one in the `HCNN MeanSimMatrix` configuration. To better study this finding, in Figure 5(*c*) we report, on the x-axis the difference in the number of features Δ_f and on y-axis the difference in the number of parameters Δ_p. As one can see, the linear relationship is strong only when the two deltas are low. For higher deltas, specifically for the three datasets "mfeat-fourier" (OpenML ID 14), "mfeat-karhunen" (OpenML ID 16), and "analcatdata_authorship" (OpenML ID 458), even if the decrement is significant for both parameters, the relationship is not linear.

Appendix H.

Table 9: For each considered model, we report the average F1_score and the corresponding standard deviation (across 10 different seeds) on the 9 benchmark datasets from the "OpenML tabular benchmark numerical classification" suite. The symbol "/" denotes the inability of the model to achieve convergence within the time or computing resources allocated for the corresponding learning task.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
361055	0.72±0.01	0.78±0.00	0.77±0.00	0.78±0.00	0.78±0.00	0.75±0.01	0.66±0.14	/	0.77±0.01	0.76±0.01
361062	0.86±0.01	0.98±0.00	0.98±0.00	0.98±0.00	0.98±0.00	0.97±0.00	0.98±0.00	/	0.99±0.00	0.99±0.00
361063	0.82±0.01	0.88±0.01	0.88±0.01	0.88±0.01	0.88±0.00	0.86±0.00	0.86±0.01	/	0.88±0.01	0.88±0.01
361065	0.77±0.01	0.86±0.00	0.86±0.01	0.86±0.01	0.86±0.00	0.84±0.01	0.86±0.01	/	0.82±0.03	0.85±0.00
361066	0.74±0.01	0.80±0.01	0.80±0.01	0.80±0.01	0.80±0.01	0.78±0.01	0.79±0.01	/	0.80±0.01	0.79±0.00
361275	0.67±0.01	0.71±0.01	0.71±0.01	0.71±0.01	0.71±0.01	0.70±0.01	0.70±0.01	/	0.70±0.01	0.70±0.01
361276	0.73±0.02	0.78±0.01	0.78±0.02	0.78±0.02	0.78±0.02	0.75±0.02	0.71±0.03	/	0.72±0.02	/
361277	0.83±0.01	0.89±0.00	0.90±0.00	0.90±0.00	0.90±0.00	0.86±0.01	0.84±0.02	/	0.88±0.01	0.88±0.00
361278	0.71±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.71±0.01	0.71±0.01	/	0.72±0.01	0.72±0.01

Table 10: For each considered model, we report the average Accuracy score and the corresponding standard deviation (across 10 different seeds) on the 9 benchmark datasets from the "OpenML tabular benchmark numerical classification" suite. The symbol "/" denotes the inability of the model to achieve convergence within the time or computing resources allocated for the corresponding learning task.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
361055	0.72±0.01	0.78±0.00	0.77±0.00	0.78±0.00	0.78±0.00	0.75±0.01	0.69±0.09	/	0.77±0.01	0.76±0.01
361062	0.86±0.01	0.98±0.00	0.98±0.00	0.98±0.00	0.98±0.00	0.97±0.00	0.98±0.00	/	0.99±0.00	0.99±0.00
361063	0.82±0.01	0.88±0.01	0.88±0.01	0.88±0.01	0.88±0.00	0.86±0.00	0.86±0.01	/	0.88±0.01	0.88±0.01
361065	0.77±0.01	0.86±0.00	0.86±0.01	0.86±0.01	0.86±0.00	0.84±0.01	0.86±0.01	/	0.82±0.03	0.85±0.00
361066	0.75±0.01	0.80±0.01	0.80±0.01	0.80±0.01	0.80±0.01	0.78±0.01	0.79±0.01	/	0.80±0.01	0.79±0.00
361275	0.67±0.01	0.71±0.01	0.71±0.01	0.71±0.01	0.71±0.01	0.70±0.01	0.70±0.01	/	0.71±0.01	0.71±0.01
361276	0.73±0.02	0.78±0.01	0.78±0.02	0.78±0.02	0.78±0.02	0.75±0.02	0.71±0.03	/	0.73±0.02	/
361277	0.83±0.01	0.89±0.00	0.90±0.00	0.90±0.00	0.90±0.00	0.86±0.01	0.84±0.02	/	0.88±0.01	0.88±0.00
361278	0.71±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.71±0.01	0.71±0.01	/	0.72±0.01	0.72±0.01

Table 11: For each considered model, we report the average MCC and the corresponding standard deviation (across 10 different seeds) on the 9 benchmark datasets from the "OpenML tabular benchmark numerical classification" suite. The symbol "/" denotes the inability of the model to achieve convergence within the time or computing resources allocated for the corresponding learning task.

Dataset ID	Model									
	LogisticRegression	**RandomForest**	**XGBoost**	**LightGBM**	**CatBoost**	**MLP**	**TabNet**	**TabPFN**	**HCNN BootstrapNet**	**HCNN MeanSimMatrix**
361055	0.45±0.03	0.56±0.01	0.55±0.01	0.55±0.01	0.55±0.00	0.50±0.02	0.41±0.15	/	0.53±0.02	0.52±0.02
361062	0.72±0.01	0.96±0.01	0.96±0.00	0.97±0.00	0.97±0.00	0.95±0.00	0.96±0.01	/	0.97±0.00	0.97±0.00
361063	0.65±0.02	0.76±0.01	0.76±0.01	0.77±0.01	0.76±0.01	0.73±0.01	0.72±0.02	/	0.75±0.01	0.75±0.01
361065	0.55±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.72±0.01	0.69±0.01	0.72±0.01	/	0.64±0.05	0.71±0.01
361066	0.49±0.01	0.60±0.01	0.60±0.02	0.60±0.01	0.60±0.02	0.56±0.02	0.57±0.01	/	0.59±0.01	0.59±0.01
361275	0.34±0.02	0.43±0.02	0.42±0.01	0.42±0.01	0.42±0.02	0.41±0.01	0.42±0.02	/	0.42±0.01	0.42±0.01
361276	0.47±0.03	0.56±0.02	0.57±0.03	0.55±0.04	0.55±0.03	0.50±0.05	0.43±0.06	/	0.45±0.04	/
361277	0.66±0.02	0.78±0.01	0.81±0.01	0.81±0.01	0.81±0.01	0.72±0.02	0.69±0.04	/	0.76±0.02	0.76±0.01
361278	0.42±0.02	0.44±0.01	0.44±0.02	0.44±0.02	0.44±0.02	0.43±0.02	0.42±0.02	/	0.43±0.02	0.44±0.02