Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Investigating over-parameterized randomized graph networks

Giovanni Donghi [a,*], Luca Pasa [a], Luca Oneto [b], Claudio Gallicchio [c], Alessio Micheli [c], Davide Anguita [b], Alessandro Sperduti [a], Nicolò Navarin [a]

[a] University of Padua, Via Trieste 63, 35121, Padova, Italy
[b] University of Genoa, Via Opera Pia 11a, 16145, Genova, Italy
[c] University of Pisa, Largo B. Pontecorvo 3, 56127, Pisa, Italy

ARTICLE INFO

Keywords:
Graph neural networks
Deep randomized neural networks
Algorithmic stability
Over-parameterization

ABSTRACT

In this paper, we investigate neural models based on graph random features for classification tasks. First, we aim to understand when over parameterization, namely generating more features than the ones necessary to interpolate, may be beneficial for the generalization abilities of the resulting models. We employ two measures: one from the algorithmic stability framework and another one based on information theory. We provide empirical evidence from several commonly adopted graph datasets showing that the considered measures, even without considering task labels, can be effective for this purpose. Additionally, we investigate whether these measures can aid in the process of hyperparameters selection. The results of our empirical analysis show that the considered measures have good correlations with the estimated generalization performance of the models with different hyperparameter configurations. Moreover, they can be used to identify good hyperparameters, achieving results comparable to the ones obtained with a classic grid search.

1. Introduction

When dealing with large-scale problems or aiming for computational efficiency, a commonly adopted approach is to exploit feature sketching (random projections followed by a component-wise non-linearity) in conjunction with a linear classifier. The behavior of linear classifiers on increasing number of random features has been studied from the theoretical point of view, in particular for ridge regression [1] and based on stochastic gradient descent [2], showing, theoretically and empirically, the presence of the double descent and best-overfit phenomena [3–7], namely the ability of these models to improve the generalization performance in over-parameterization, i.e., when having much more parameters than the ones needed to interpolate. To the best of authors' knowledge, no study in literature considers the case of random graph neural features. We speculate that the reason is that the majority of randomized graph networks in literature are based on a recurrent scheme of Reservoir Computing that, while generating expressive features, can result in a high computational complexity with hundreds or thousands of features [8].

However, recently an untrained graph neural model has been proposed [9] that can efficiently generate thousands of non-linear features, only specifying the number of desired features and a scale parameter controlling the magnitude of the randomly generated weights. The computational efficiency of this untrained graph model allows us to investigate the generated representations.

The first goal of this paper is to extend the study by Navarin et al. [10] of how over-parameterization impacts learning on graph domains exploiting randomized graph neural networks. The authors observed that, in the cases in which over-parameterization is beneficial, a metric derived from Algorithmic Stability, the condition number of the Gram matrix of the hidden representations [3,4], shows a distinct behavior.

The second contribution of this paper is to exploit different indicators of representational richness to understand if the over-parameterization regime can be beneficial for the task defined on a given dataset or not. Specifically, since in some instances the computation of the condition number is not reliable, we introduce an additional measure based on information theory, that is the average feature entropy [11], whose computation is less prone to numerical instabilities.

As a third contribution, we study the relationship between the aforementioned measures and the estimated generalization performance of the model, showing generally a good correlation when varying both the number of generated features and the scale parameter. Moreover, we

* Corresponding author.
E-mail addresses: giovanni.donghi@phd.unipd.it (G. Donghi), luca.pasa@unipd.it (L. Pasa), luca.oneto@unige.it (L. Oneto), claudio.gallicchio@unipi.it (C. Gallicchio), alessio.micheli@unipi.it (A. Micheli), davide.anguita@unige.it (D. Anguita), alessandro.sperduti@unipd.it (A. Sperduti), nicolo.navarin@unipd.it (N. Navarin).

https://doi.org/10.1016/j.neucom.2024.128281
Received 27 February 2024; Received in revised form 21 May 2024; Accepted 24 July 2024
Available online 7 August 2024

show that the considered metrics can be used to perform model selection, achieving in most cases results comparable to the ones obtained by a classic grid search approach, yet without the need to train but the selected model.

2. Background and related works

In the forthcoming discussion, we will use the following notation: we refer to variables with italic letters, to vectors with bold lowercase letters, to matrices with bold uppercase letters, and to sets or tuples with uppercase calligraphic letters. The elements of a matrix or vector are indicated with lowercase italic and appropriate indices, such as a_{ij} to refer to the entries of matrix $\mathbf{A}$.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph, where $\mathcal{V} = \{v_1, \dots, v_n\}$ denotes the set of vertices (nodes) of the graph, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} \in \mathbb{R}^{n \times s}$ are node attributes, with the ith row representing the s-dimensional feature vector $\mathbf{x}_i$ of v_i. We indicate the adjacency matrix of the graph with $\mathbf{A} \in \mathbb{R}^{n \times n}$, with elements $a_{ij} = 1 \iff (v_i, v_j) \in \mathcal{E}$.

2.1. Randomized graph neural networks

In recent years, there has been a noticeable trend towards increasingly complex models within structured data domains. These models have been pushing the boundaries of architecture design, resulting in novel frameworks characterized by a substantial number of parameters. However, this surge in complexity comes at a cost, particularly in terms of computational resources required for training these models. Within the sequential data domain, efficient architectures to address similar issues have been developed using the Reservoir Computing (RC) paradigm [12]. RC exploits the Echo State Property (ESP) [13] to randomize the values of recurrent parameters ensuring stability conditions. As such, Echo State Networks (ESN) [13] are commonly and efficiently employed for sequential data. Given the reliance on untrained dynamics of the reservoir, the richness of the resulting neural representations has been assessed using metrics such as the average state entropy [11].

Authors of [14] were the first to propose to exploit the reservoir computing (randomized) framework for graph structured data as well. They proposed GraphESN, a model comprised of a non-linear reservoir for graph embedding and a feed-forward linear readout. The global state is computed by an iterative process, in which the reservoir performs a fixed recurrent function until convergence. Fast and Deep GNN (FDGNN) [15] was later introduced as and evolution of GraphESN, and leverages stacked layers of GNN as a reservoir, constructing progressively more general graph features. Authors of [16] proposed a model, the Multi-resolution Reservoir Graph Neural Network (MRGNN), which exploits a Reservoir Convolutional layer for graphs able to simultaneously and directly consider all topological receptive fields up to k-hops. More recently, authors of [17] focused on the task of node classification using randomized graph convolutions (differently from this paper in which we consider the more challenging setting of graph classification). Contrary to many graph neural networks, the authors proposed a single-layer architecture, yet with an increased receptive field, defined as $\mathbf{Z} = \sigma(\mathbf{A}^2\mathbf{X}\mathbf{W})\boldsymbol{\beta}$, where σ is the sigmoid function, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of the graph, $\mathbf{W} \in \mathbb{R}^{s \times m}$ is the (random) wight matrix for m hidden neurons (that is left untrained), and $\boldsymbol{\beta}$ are the trained output weights.

2.2. Over-parameterization

Simultaneously, deep learning researchers have started to question themselves about the mechanism of over-parameterization, which, despite its longstanding study [18], has recently revealed profound implications for advanced deep networks such as large language models [19]. In essence, over-parameterization involves leveraging models with more parameters than the ones necessary to interpolate the data, namely perfectly fit or memorize the available data, to obtain good generalization performance [3]. Recent findings showed that, even if counter-intuitive, increasing the number of parameters after the interpolating threshold can enhance the generalization ability of the model by increasing its stability [3,4]. Unfortunately, stability is not always simple and straightforward to compute in order to evaluate the generalization ability of a model [3]. As we motivate in Section 4.2, in this work we compute stability via the condition number of the Gram matrix [3–5] induced by the random graph projection.

3. Graph random features

The primary aim of this paper is to investigate the effectiveness of over-parameterized graph models using graph random features. In order to understand if and under which conditions such models are feasible and convenient, we leverage measures from statistical learning theory, such as the Algorithmic Stability, and from information theory, such as entropy. In this section we first introduce a summary of a recent approach for generating graph random features based on graph neural networks [9], which we use for the present study, and then we describe the usage of entropy to measure the richness of the extracted representations. Afterwards, in Section 4.2, we will delve deeper into statistical learning theory and approximations of Algorithmic Stability.

3.1. Feature extraction

The architecture for graph random feature extraction introduced by the authors of [9], dubbed U-GCN, is closely inspired by fully trained graph neural networks, with the difference of not being trained. Specifically, multiple graph convolution layers are stacked one after the other, interleaved with a hyperbolic tangent element-wise non-linear activation function. For the definition of graph convolution, GCN [20] is used, resulting in hidden node representations computed by the lth layer as $\mathbf{H}^{(l)} = tanh(\mathbf{S}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)})$, where $\mathbf{S} = (\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})$ is the normalized Laplacian adopted by the GCN ($\tilde{\mathbf{A}} = \mathbf{A} + \mathbb{I}$ is the adjacency matrix with the addition of self loops, and $\tilde{\mathbf{D}}$ is its diagonal degree matrix where $\tilde{d_{ii}} = \sum_j a_{ij} + 1$), $\mathbf{W}^{(l)}$ are the layer parameters and $\mathbf{H}^{(0)} = \mathbf{X}$. For the sake of simplicity, in our notation we omit the bias terms. Crucially for the graph random features approach, the weight in $\mathbf{W}^{(l)}$ are randomly initialized by the Glorot uniform approach [21] with a gain hyperparameter θ to control the effective scaling of $\mathbf{W}^{(l)}$, and successively left untrained. As a result, a weight matrix of shape $n \times m$ will have entries sampled from a uniform distribution $\mathcal{U}(-a, a)$ where $a = \theta\sqrt{6/n+m}$. Finally, node representations are obtained by concatenating the representations computed by each layer of the network, i.e. $\mathbf{H} = [\mathbf{H}^{(1)}, \dots, \mathbf{H}^{(L)}]$, where L is the number of layers of the network.

3.2. Global pooling

The untrained graph convolutional network we described produces node representations that capture local information through the neighborhood aggregation of graph convolution. Thus, to perform graph-level tasks, a *global pooling* layer is exploited to obtain a single representation for the whole graph. Commonly used global pooling operators include maximum, minimum and average pooling and are generally required to be differentiable to be used for end-to-end learning by backpropagation. In the case of untrained graph convolution, since we do not require gradients, non-differentiable operators can be utilized as well, such as the Percentage of Positive Values (PPV) [22]. The PPV is a non-differentiable pooling mechanism introduced for randomized networks for time series and defined as: $PPV(\mathbf{z}) = \frac{1}{n}\sum_{i=1}^{n} I[z_i > 0]$, where $I[z_i > 0]$ is the indicator function whose value is 1 if $z_i > 0$, 0 otherwise. The authors proposed using jointly the Global Max Pooling and PPV as global pooling, concatenating the resulting representations. Note that this choice doubles the size of the global graph representation compared to the representations of the single nodes provided in output by the untrained graph convolution. Finally, the authors proposed to use a ridge classifier as a readout for its computational efficiency.

4. Richness of representations

4.1. Average feature entropy

In the context of graph random features, a fundamental concern revolves around the diversity and expressiveness of the representations they capture. Given their random and untrained nature, it is crucial to assess their effectiveness in encapsulating various graph structures and patterns. To quantify this, we rely on a suitably adapted average state entropy (ASE), a metric that has been used to assess the richness of dynamics in reservoir computing [11,23,24].

Considering reservoir layers with fixed number of units u and a given sequence, the instantaneous state entropy, computed with efficient estimator of Renyi's quadratic entropy [25], for each time-step t and layer l is defined as

$$H^{(l)}(t) = -\log\left(\frac{1}{u^2}\sum_{j=1}^{u}\left(\sum_{k=1}^{u}\mathcal{K}\left(h_j^{(l)}(t) - h_k^{(l)}(t)\right)\right)\right), \tag{1}$$

where $h_j^{(l)}(t)$ is the jth component of the reservoir state $\mathbf{h}^{(l)}(t)$ and $\mathcal{K}$ is a Gaussian kernel. Given an input time-series of length T, the ASE is then defined as

$$ASE^{(l)} = \frac{1}{T}\sum_{t=1}^{T} H^{(l)}(t) \tag{2}$$

With this definition, ASE quantifies the richness of each deep reservoir layer, averaged across the length of the time-series. To adapt it to our setting, we first observe how an entropy akin to Eq. (1) is sufficient to capture the richness of a graph representation within a layer, since we do not have a time dimension. Then, as we concatenate the representations of all the layers to obtain the graph features, we take the average of the entropy of the layers to quantify the richness of the entire graph representation. This entropy reflects the diversity of node representations within the layers: intuitively, more diverse representations can provide richer features from which the successive layers and the trainable readout can extract relevant patterns. Finally, to obtain an average measure of the quality of graph features for a particular dataset, we calculate average feature entropy (AFE) by taking the mean over all the graphs it contains.

More formally, considering a dataset of m graphs $\mathcal{D}_m = \{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$, the entropy of a graph's random features is computed for graph G_i at each layer l as

$$H^{(l)}(\mathcal{G}_i) = -\log\left(\frac{1}{u^2}\sum_{j=1}^{u}\left(\sum_{k=1}^{u}\mathcal{K}\left(h_j^{(l)}(\mathcal{G}_i) - h_k^{(l)}(\mathcal{G}_i)\right)\right)\right), \tag{3}$$

where $h_j^{(l)}(\mathcal{G}_i)$ is the jth component of the hidden representation of $\mathcal{G}_i$ at layer l, u is the number of units in layer l, and $\mathcal{K}$ is a Gaussian kernel.[1] The feature entropy of $\mathcal{G}_i$ is then calculated as the average of all layers as

$$H(\mathcal{G}_i) = \frac{1}{L}\sum_{l=1}^{L} H^{(l)}(\mathcal{G}_i). \tag{4}$$

Finally, average feature entropy is obtained by taking the mean of graph feature entropy over all graphs in the training dataset:

$$AFE_{\mathcal{D}_m} = \frac{1}{m}\sum_{i=1}^{m} H(\mathcal{G}_i). \tag{5}$$

[1] In particular, the size of the Gaussian kernel $\mathcal{K}$ is set to 0.3 times the standard deviation of unit activations for the given layer, similarly to [11].

4.2. Hypothesis stability

Let us consider the supervised learning setting [26–28]. Based on a random observation of $X \in \mathcal{X}$ one has to estimate $Y \in \mathcal{Y}$ sampled according to μ over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ by choosing a suitable function $f : \mathcal{X} \to \hat{\mathcal{Y}}$ in a set $\mathcal{F}$. A learning algorithm $\mathscr{A}_\omega$, characterized by hyperparameters ω, selects $\hat{f} \in \mathcal{F}$, by exploiting a set of m labeled samples $\mathcal{D}_m = \left\{(X_1, Y_1), \ldots, (X_m, Y_m)\right\} = \left\{Z_1, \ldots, Z_m\right\}$ with $Z \in \mathcal{Z}$ sampled independently from μ. Formally

$$\hat{f} = \mathscr{A}_\omega(\mathcal{D}_m). \tag{6}$$

The generalization error of $\hat{f}$, i.e., the error of $\hat{f}$ in approximating $\mathbb{P}\{Y|X\}$, is defined as

$$\mathsf{L}(\hat{f}) = \mathbb{E}_Z\left\{\ell(\hat{f}, Z)\right\}, \tag{7}$$

where $\ell : \mathcal{F}\times\mathcal{Z} \to [0,1]$ is a loss function which measures the pointwise quality of the approximation. Since $\mathsf{L}(\hat{f})$ cannot be explicitly computed, we can compute the empirical error [26], i.e., the empirical risk or training error

$$\hat{\mathsf{L}}(\mathscr{A}_\omega(\mathcal{D}_m)) = \hat{\mathsf{L}}_{\text{emp}}(\mathscr{A}_\omega(\mathcal{D}_m)) = \frac{1}{n}\sum_{Z\in\mathcal{D}_m}\ell(\mathscr{A}_\omega(\mathcal{D}_m), Z). \tag{8}$$

Another possible estimator is the leave one out error [29]

$$\hat{\mathsf{L}}_{\text{loo}}(\mathscr{A}_\omega(\mathcal{D}_m)) = \frac{1}{m}\sum_{Z\in\mathcal{D}_m}\ell(\mathscr{A}_\omega(\mathcal{D}_m \setminus Z), Z), \tag{9}$$

namely the average error on a single sample of $\mathcal{D}_m$ kept away from the learning phase.

In this setting it is possible to prove that [30]

$$\mathbb{P}\left\{\mathsf{L}(\mathscr{A}_\omega(\mathcal{D}_m)) \le \hat{\mathsf{L}}_*(\mathscr{A}_\omega(\mathcal{D}_m)) + \mathsf{M}(\mathscr{A}_\omega) + \Delta(m,\delta)\right\} \ge 1-\delta, \tag{10}$$

namely, the generalization error of $\hat{f}$ is bounded by one of its empirical estimator $\hat{\mathsf{L}}_* \in \{\hat{\mathsf{L}}_{\text{emp}}, \hat{\mathsf{L}}_{\text{loo}}\}$ plus a term $\mathsf{M}(\mathscr{A}_\omega)$ which measures the risk due to the choice of the algorithm and its hyperparameters (i.e., the more the algorithm tends to memorize/fit/extract-information and not learn from the data the larger is this term), plus a confidence term[2] $\Delta(n,\delta)$ which measures the risk associated to the sample (i.e., the less data we have or the larger confidence we require, the larger is this term).

A very effective way to estimate $\mathsf{M}(\mathscr{A}_\omega)$ is the one based on Algorithmic Stability (AS) [28–35]. AS allowed multiple times to shed new light on generalization [32] or on more complex phenomena like overparameterization [3]. The underlying idea of AS is actually simple: the more accurate the model chosen by the algorithm is and the more similar are the models chosen when the training data are (slightly) modified then the more the algorithm generalizes.

Many different notions of stability exist [4,29,31,34], e.g., uniform stability, hypothesis stability (HS), cross validation and leave one out stability, error stability, and pointwise HS. Nevertheless, the one that has been shown to be the most important/powerful/insightful to understand complex phenomena is the HS. HS, while inducing looser bounds with respect to, e.g., uniform stability [29–31], can be estimated from the data and it is strongly dependent on the properties of the algorithm [3,4,29,30,34]. HS can be defined in two different ways [29,36,37]

$$\mathbb{E}_{\mathcal{D}_m,Z_i}\left|\ell(\mathscr{A}_\omega(\mathcal{D}_m), Z_i) - \ell(\mathscr{A}_\omega((\mathcal{D}_m \setminus Z_i)\cup Z_i'), Z_i)\right| \le \beta_{\text{emp}}, \tag{11}$$

$$\mathbb{E}_{\mathcal{D}_m,Z'}\left|\ell(\mathscr{A}_\omega(\mathcal{D}_m), Z') - \ell(\mathscr{A}_\omega(\mathcal{D}_m \setminus Z_i), Z')\right| \le \beta_{\text{loo}}, \tag{12}$$

where Z' is a sample sampled according to μ. Then, if we use $\hat{\mathsf{L}}_{\text{emp}}$ in Bound (10) we have that $\mathsf{M}(\mathscr{A}_\omega) \propto \beta_{\text{emp}}$,while if we use $\hat{\mathsf{L}}_{\text{loo}}$ in Bound (10) we have that $\mathsf{M}(\mathscr{A}_\omega) \propto \beta_{\text{loo}}$.

[2] We will not discuss this term since it is independent from $\mathscr{A}_\omega$.

Note also that if ℓ is L-Lipschitz continuous with respect to a notion of distance $\mathtt{d}(\cdot,\cdot)$ where $\mathtt{d}: \hat{\mathcal{Y}} \times \hat{\mathcal{Y}} \to \mathbb{R}$ we can state that

$$\beta_{\text{emp}} \leq L\ \mathbb{E}_{\mathcal{D}_m, X_i} \mathtt{d}(\mathscr{A}_\omega(\mathcal{D}_m)(X_i), \mathscr{A}_\omega(\mathcal{D}_m \setminus Z_i \cup Z_i')(X_i)), \tag{13}$$

$$\beta_{\text{loo}} \leq L\ \mathbb{E}_{\mathcal{D}_m, X'} \mathtt{d}(\mathscr{A}_\omega(\mathcal{D}_m)(X'), \mathscr{A}_\omega(\mathcal{D}_m \setminus Z_i)(X')). \tag{14}$$

These formulations of the HS are quite useful since they can be estimated in a fully unsupervised way once the models have been trained. This is quite useful when we do not simply want to estimate the stability of the end-to-end trained model but, for example, we want also to estimate the stability of the different representation layers of a deep model. In fact, after $\hat{f} = \mathscr{A}_\omega(\mathcal{D}_m)$ has been trained, we can change the target $\hat{\mathcal{Y}}$ of the $\hat{f}$ into a portion of the inner representation which generates $\hat{f}$ and, provided a notion of distance $\mathtt{d}(\cdot,\cdot)$ that can be applied to them, the quantities of Eqs. (13) and (14) can be computed.

In fact, note that the quantities of Eqs. (11)–(14) can be easily estimated both with practical approaches (e.g., Bootstrap [38] or Bag of Little Bootstrap [39] if we want to reduce the computational effort) or with theoretical approaches [3,30] obtaining $\hat{\beta}_{\text{emp}}$ and $\hat{\beta}_{\text{loo}}$.

Moreover, in some specific cases, it is possible to bound, instead of estimating, the HS [3,4]. Specifically, let us consider that f can be expressed as $f(X) = \boldsymbol{w}\boldsymbol{\phi}(X)$, where $\boldsymbol{\phi}: \mathcal{X} \to \mathbb{R}^D$ and $\boldsymbol{w} \in \mathbb{R}^D$, then

$$\beta_{\text{loo}}, \beta_{\text{emp}} \propto \mathsf{Cond}(\mathbf{HH}'), \tag{15}$$

where $\mathbf{H} = [\boldsymbol{\phi}(X_1), \dots, \boldsymbol{\phi}(X_n)]'$, $\mathbf{HH}'$ is the Gram matrix, and Cond indicates the condition number (the ratio between the largest and smallest singular values). Note also that, the same property holds when we consider inner layer of the representation $\boldsymbol{\phi}(X)$ as we discussed for the quantities of Eqs. (13) and (14). In the context of our analysis, matrix $\mathbf{H}$ collects the representations with random features of the graphs, and we therefore use the condition number of the Gram matrix $\mathbf{HH}'$ for HS.

5. Experimental results

In this section, we illustrate our experimental setting[3] and we present evidence of the fact that Algorithmic Stability, through the condition number, and the richness of graph representations, measured as average feature entropy, are able to provide useful insights about the effectiveness of over-parametrization for improving the generalization abilities of neural models based on graph random features. After presenting the considered datasets and the experimental setup, we will show how the two aforementioned techniques behave in under- and over-parameterized settings on the considered datasets and show some consistent behavior of the metrics that we observed. After that, we study how the proposed metrics can be exploited not only for the choice of over- or under- parameterization of the network, but for model selection as an alternative to grid-search with performance estimation. Our results are encouraging, showing that in many settings the metric values have strong correlation with the measured generalization error.

5.1. Datasets

In order to evaluate empirically the extraction of graph random features, we considered seven commonly adopted benchmark datasets for graph classification from the TUDatasets collection [40], five pertaining to bioinformatics and two social network datasets. DD [41], ENZYMES [41] and PROTEINS [41] represent proteins as graphs with secondary structures, such as amino acids as nodes, annotated with physical and chemical information. Nodes are connected by an edge if they are either neighbors in the amico acid sequence or are less than 6 Å apart. The task on ENZYMES consists in classifying 6 high-level classes of enzymes, and for DD and PROTEINS it is to predict if a given protein is an enzyme. NCI1 [42] and PTC_MR [43] consist of molecule graphs, where each node represents an atom, with labels encoding atom type, and edges given by chemical bonds. The task here is to predict whether molecules are carcinogenic or not. Finally, we also use two social network datasets, IMDB-BINARY and IMDB-MULTI [44], which contain ego-networks with information about actors/actresses linked by collaborations, with the task of predicting the genre of movies in which they act. Some summary statistics of these datasets are reported in Table 1.

Table 1
Statistics of the datasets used for the experiments.

Dataset	# classes	# graphs	avg. # nodes	avg. # edges
DD	2	1178	284.32	715.66
ENZYMES	6	600	32.63	62.14
PROTEINS	2	1113	39.06	72.82
NCI1	2	4110	29.87	32.30
PTC_MR	2	344	14.29	14.69
IMDB-BINARY	2	1000	19.77	96.53
IMDB-MULTI	3	1500	13.00	65.94

5.2. Experimental setup

We study the behavior of the model described in Section 3 varying the number of units in the graph convolutional layers and the gain hyperparameter θ. As the purpose of this paper is to investigate the impact of over-parameterization and how to analyze it through the metrics of entropy and stability, rather than to develop a model with state of the art performance, we use the same four layer configuration of U-GCN, whose performance was shown to be competitive with the state of the art [9]. The number of units for each layer u spanned from 10 to 10,000 per layer, in the set $\{10, 20, 30, 50, 75, 100, 250, 500, 1000, 2000, 3000, 5000, 7500, 10000\}$. Since we use four layers and concatenate two different readouts, the resulting graph representation is up to size 80,000. However, since the weights are not trained, we just have to perform the forward phase which is extremely fast even with a high number of features to extract. Then, we trained a ridge classifier characterized by a regularization hyperparameter α taking values in the set $\{0, 10^{-4}, \dots, 10^5\}$. We also considered multiple values of θ, the gain for weight initialization, in the set $\{0.01, 0.1, 0.5, 1, 5/3, 3, 5, 10, 30, 50\}$. Following the validation procedure of [45], we estimate the performance and metrics of the untrained models by performing 5-fold cross-validation, repeating the whole procedure 5 times to account for the random initialization.

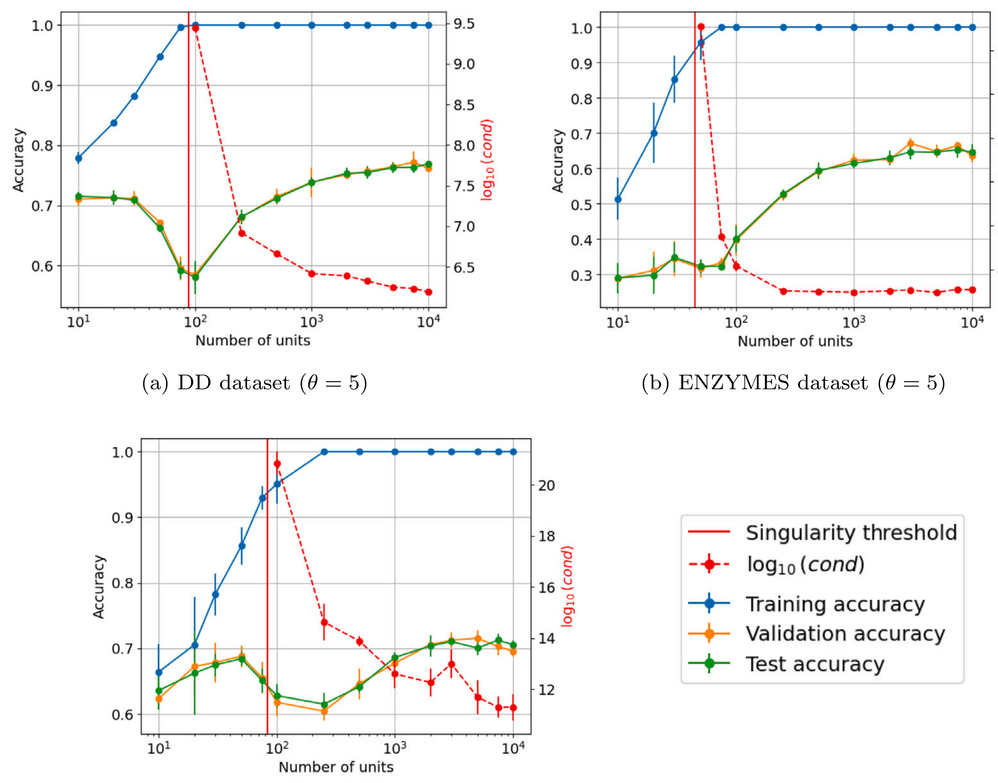
5.3. Metrics calculation

Entropy
Due to the high computational cost of evaluating Eq. (3) for all the graphs in the training set for all combinations of hyperparameters, we computed an unbiased estimator of the entropy in Eq. (5) by averaging over a subset (10%) of the graphs.

Conditioning
Given the matrix $\mathbf{H}$ collecting representations of all training graphs, we computed the condition number of the gram matrix $\mathbf{HH}'$. Given the high values of such condition numbers, we used equality $\mathsf{Cond}(\mathbf{HH}') = \mathsf{Cond}(\mathbf{H})^2$ to check for numerical problems. When this inequality is not satisfied (at least within an order of magnitude), we consider the resulting condition number too inaccurate due to the numerical issues. We used increased precision (up to octuple precision, using the Advanpix[4] Multiprecision Computing Toolbox for MATLAB) for this

[3] Code: https://github.com/giovannidonghi/overparam-random-graph-nets.

[4] https://www.advanpix.com/



(a) DD dataset ($\theta = 5$)

(b) ENZYMES dataset ($\theta = 5$)

(c) PROTEINS dataset ($\theta = 50$)

Fig. 1. Results showing the logarithm of the condition number together with training, validation and test accuracy for regularization hyperparameter $\alpha = 0$, fixed value of gain hyperparameter θ (shown in brackets) and varying number of units per layer. Error bars indicate standard deviation across runs. The singularity threshold is the minimum number of units per layer required for the Gram matrix $\mathbf{HH}'$ to not be singular.

sanity check. Only for the DD, ENZYMES and PROTEINS datasets the sanity check was satisfied, while for the other datasets the condition number continued to increase by increasing the precision, thus it was considered unreliable due to the numerical instability. Therefore, these cases (indicated with N/A in tables and not shown in plots) were discarded to provide a reliable analysis. Furthermore, since the rank of $\mathbf{HH}'$ is necessarily lower than or equal to the number of features, we compute its condition number only after the singularity threshold, where the number of features equals the number of training graphs.

5.4. Results

In order to keep the reading from becoming too heavy, only a subset, the most informative, of the plots is reported and discussed, with additional ones available as supplementary material, on which similar considerations hold. We report in Fig. 1 the results of some configurations that reach competitive performance for DD, ENZYMES and PROTEINS datasets. We report Algorithmic Stability estimated via the condition number of the Gram matrix, together with accuracies achieved on training, validation and test sets, at varying number of units. Similarly, in Fig. 2, we report the results for average feature entropy with the same configurations, with the addition of plots for IMDB-BINARY and IMDB-MULTI.

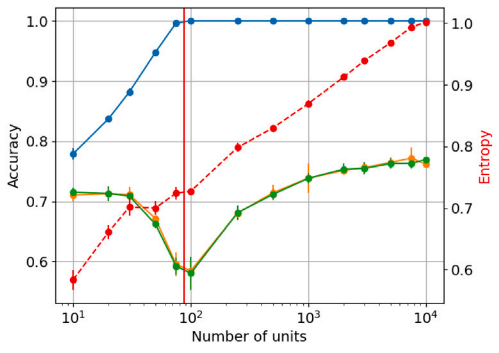
For a more general and comprehensive visualization of the results, we show the values of the different metrics varying both hyperparameters in Fig. 3, alongside the performance on the test set with validated regularization hyperparameter α. The condition number is computed and shown only after the singularity threshold.

Other than the qualitative assessment given by visual inspection of the plots in Figs. 1–3, we want to understand more generally how good Algorithmic Stability and entropy are as indicators of model

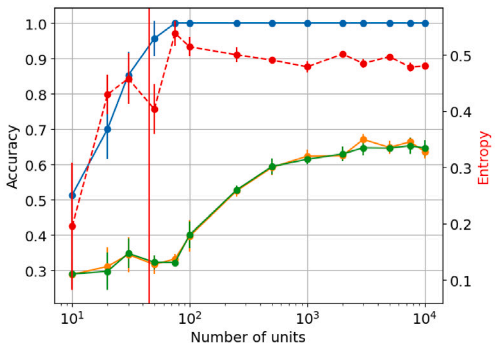
Table 2
Spearman's rank correlation coefficient between metrics ("Cond" stands for the condition number of the Gram matrix) and test performance, both with validated regularization hyperparameter α and without regularization. Correlations are marked based on p-value, with * if $p < 0.05$, ** if $p < 0.01$ and *** if $p < 0.001$. Negative correlation is best for condition number, positive correlation is best for entropy. We indicate the cases where the computation of the condition number was unstable, as described in Section 5.3, with N/A.

Dataset	Spearman's correlation with performance			
	Cond		Entropy	
	valid. α	$\alpha = 0$	valid. α	$\alpha = 0$
DD	-0.264^{*}	-0.711^{***}	0.764^{***}	0.441^{***}
ENZYMES	-0.451^{***}	-0.541^{***}	0.650^{***}	0.615^{***}
PROTEINS	-0.231^{*}	-0.589^{***}	0.400^{***}	-0.195^{*}
NCI1	N/A	N/A	0.433^{***}	0.445^{***}
PTC_MR	N/A	N/A	0.222^{**}	0.270^{**}
IMDB-BINARY	N/A	N/A	0.446^{***}	0.367^{***}
IMDB-MULTI	N/A	N/A	0.210^{*}	-0.138

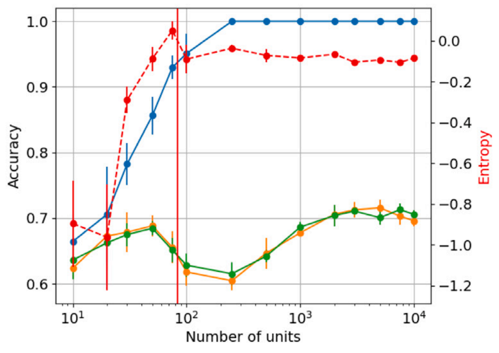
performance, and if they can be used to identify good hyperparameter without requiring supervision. We thus propose two quantitative approaches to assess the quality of the proposed metrics as proxies for the generalization capacity of the models: we compute Spearman's rank correlations between metrics and performance, and we measure the performance gap between optimal choice of hyperparameters with model selection on the validation set and the metric-driven choice. We favored Spearman's rank correlation, which measures monotonicity and not linearity, instead of Pearson's correlation as we observed that in some cases the relationships were not linear. For each dataset, Spearman's rank correlation between metric and test performance is calculated over all pairs of hyperparameters, gain θ and number of hidden units u, to obtain an indication of monotonicity, and therefore



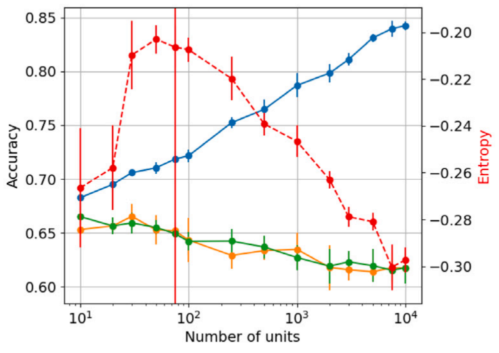
(a) DD dataset ($\theta = 5$)



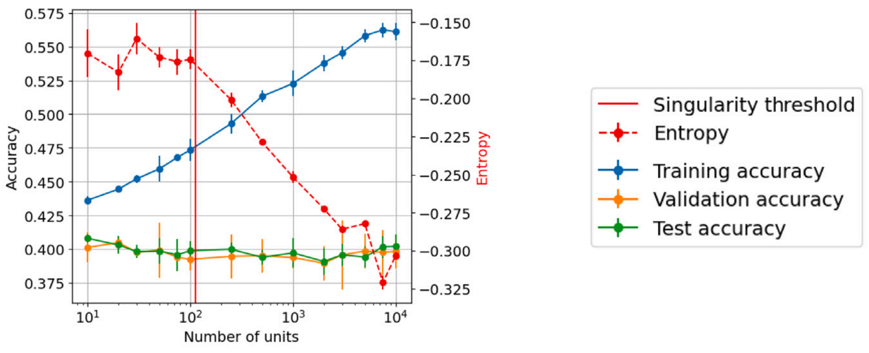
(b) ENZYMES dataset ($\theta = 5$)



(c) PROTEINS dataset ($\theta = 50$)



(d) IMDB-BINARY ($\theta = 5$)



(e) IMDB-MULTI ($\theta = 5$)

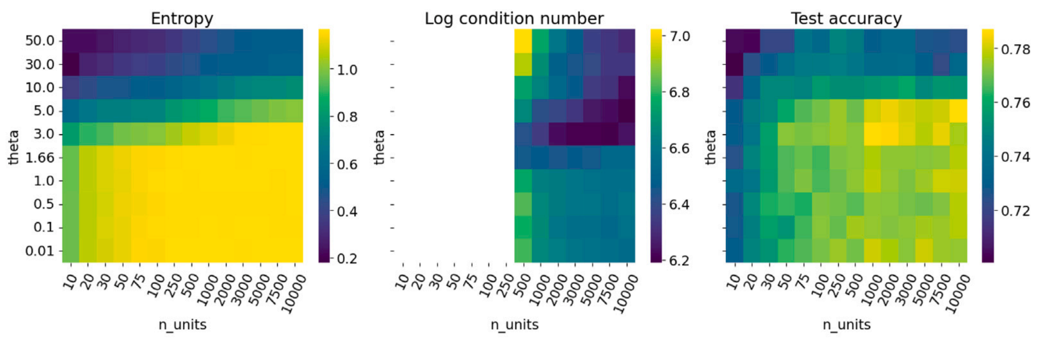
Fig. 2. Results showing the entropy together with training, validation and test accuracy for regularization hyperparameter $\alpha = 0$, fixed value of gain hyperparameter θ (shown in brackets) and varying number of units per layer. Error bars indicate standard deviation across runs. The singularity threshold is the minimum number of units per layer required for the Gram matrix $\mathbf{HH}'$ to not be singular.

of how closely the two metrics follow the performance on the test set (thus the generalization). Correlations with performance, both with validated regularization hyperparameter α and without regularization, are reported in Table 2, together with the significance level of the t-test for Spearman's rank correlation. The relationships between metrics and performance are also visualized in Fig. 4. Additional plots and Pearson's correlations are available in the supplementary materials.

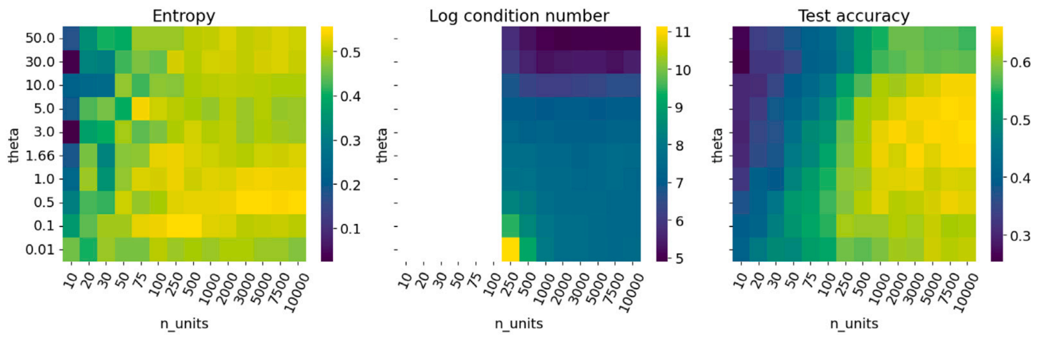
While correlation gives us an intuition of the overall trend of performance and metrics, we would also like to know if we can successfully use the metrics as proxies of performance. We want to measure how close to the optimum we can get by only considering the given metric of the graph representations, without even training multiple classifiers and keeping the best performing one. For this assessment we consider the margin between the optimal choice of hyperparameters with standard model selection (the best performing pair of hyperparameters gain θ and number of units per layer u on the validation set), and the pair that would be selected by the given metric (with lowest condition number or highest entropy). For this case as well we considered both validated regularization hyperparameter α and no regularization at all, reporting the observed results in Table 3.

6. Discussion

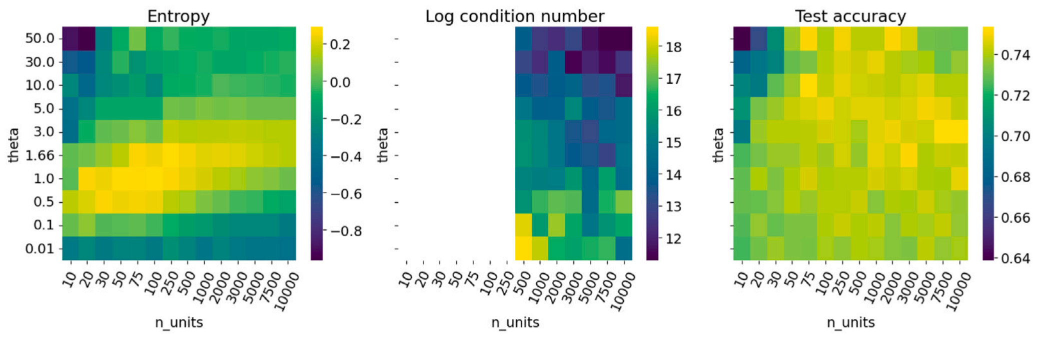
From the results of Fig. 1, of the models' performance without regularization, we clearly observe the phenomenon of double-descent: on DD and PROTEINS in particular, we observe how accuracy on the test set drops quite significantly around the interpolation threshold, yet on all three datasets once we increase the number of units into the over-parameterized regime performance increases. The condition number of the Gram matrix mirrors this trend, with very high values just over the singularity threshold, and then it generally continues to decrease with a higher number of units. Thus, Algorithmic Stability shows us that, in over-parameterized regimes, adding more neurons can enhance generalization rather than hinder it.



(a) DD dataset



(b) ENZYMES dataset



(c) PROTEINS dataset

Fig. 3. Entropy (the higher the better), logarithm of the condition number of Gram matrix (the lower the better) and performance measured with accuracy (the higher the better) with validated regularization hyperparameter α, on DD, ENZYMES and PROTEINS datasets, at varying number of hidden units per layer u (horizontal axis) and gain hyperparameter θ (vertical axis). The condition number is computed only after the singularity threshold, and it is shown accordingly.

Observing the results of average feature entropy in Fig. 2, we can see that the richness of the representations can give us similar insights to the condition number on DD, ENZYMES and PROTEINS datasets: on ENZYMES and PROTEINS we see entropy clearly distinguishing the under- and over-parameterized regimes, while on DD there is a steady growth with the increase of number of units. In these cases, therefore, entropy captures the increase in the richness of representations that over-parameterization allows, which is reflected in the generalization of the model. We report also the results on the IMDB datasets in Figs. 2(d) and 2(e), where we observe a different behavior. In this case, the model struggles to even fit the training data, even after reaching the singularity threshold. Additionally, test performance either does not improve or even worsens as the number of units increases. These results suggest that on the IMDB datasets over-parameterization is not beneficial for generalization, and we can see that entropy supports this, showing a decreasing trend. Average feature entropy is thus informative also in cases in which over-parameterization is not successful, where the condition number might instead be too high to be calculated robustly.

From Fig. 3 we can draw some more general observations on the DD, ENZYMES and PROTEINS dataset. First, we observe how the accuracy on the test set generally increases after the interpolation/singularity threshold (which is not explicitly reported but can be seen from the presence of the condition number). Thus the use of graph random features appears to benefit from an over-parameterized regime. While without the use of regularization in Fig. 1 we clearly observed double-descent, in this case the use of regularization mitigates the phenomenon. Algorithmic Stability, as measured by the condition number of the Gram matrix, and the average feature entropy, as a measure of the richness of representations, are able to tell us that overall adding more neurons can improve generalization instead of hurting it, as can also be seen from the plots in Fig. 4 thanks to the color indicating the number of hidden units. In some cases, such as with the DD dataset, we can see a striking similarity between the entropy and the achieved performance, and a region in the hyperparameter space with lowest condition number overlaps very well with the best generalization results.

The more quantitative results of Table 2 give further supporting evidence. We observe very good and significant negative correlation between the logarithm of the condition number and performance without regularization, while when $\alpha \neq 0$ entropy seems to be the best

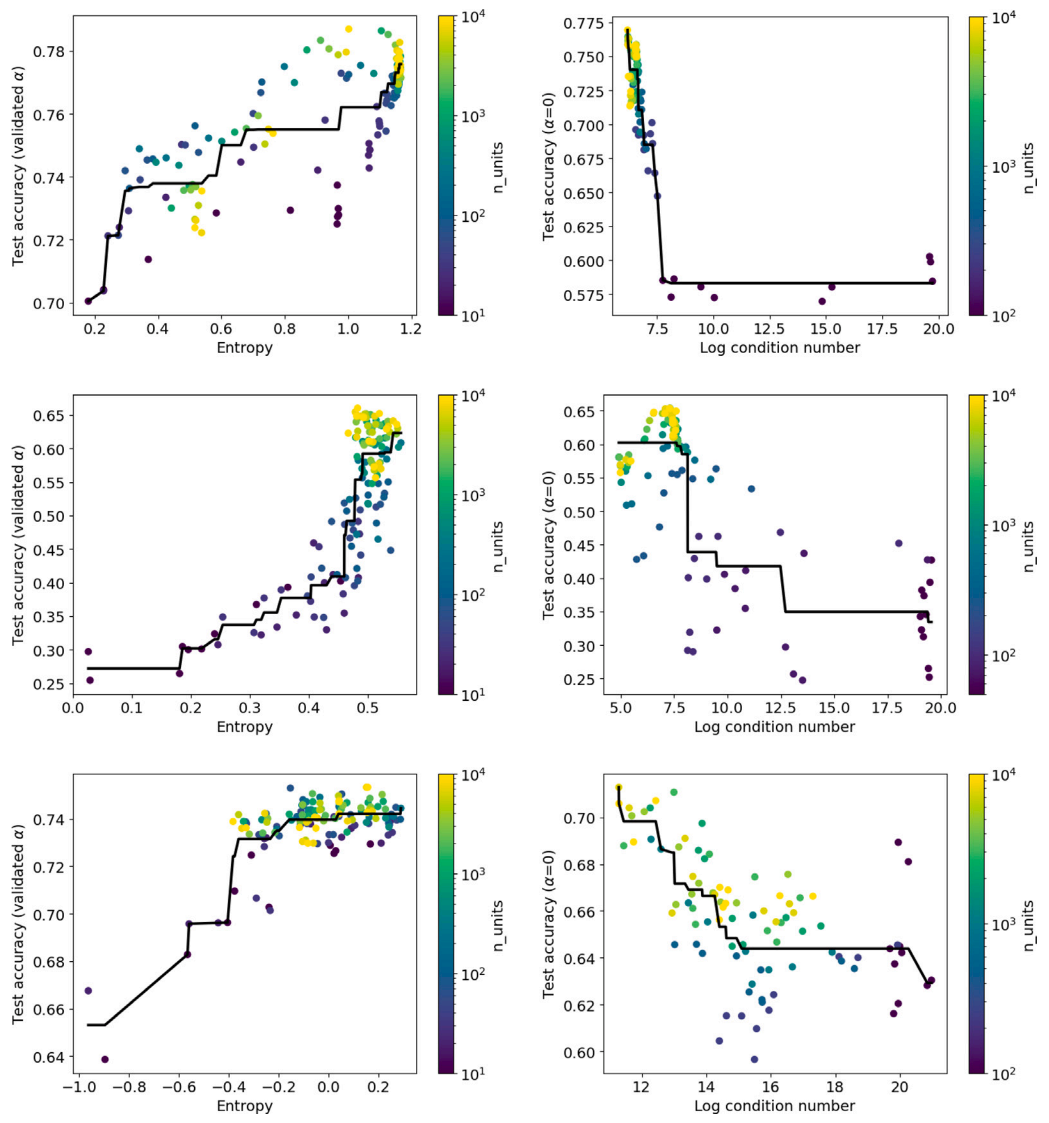


Fig. 4. Visualizations of the relationship between performance with entropy (left column) and logarithm of the condition number (right column), for DD, ENZYMES and PROTEINS datasets. Entropy is correlated to performance with regularization, the condition number to performance without regularization (complementary plots available in supplementary materials). For each pair of hyperparameters, gain θ and number of hidden units u, we plot the respective measures, with color indicating u. We additionally plot the isotonic regression line to observe the measure of monotonicity captured by Spearman's rank correlation.

Table 3
Comparison of test performance (accuracy) with different model selection: standard model selection using validation performance (Val) and using the two metrics to select hyperparameters ("Cond" stands for the condition number of the Gram matrix). Accuracy is reported with standard deviation over 5 runs. We report results both with validated regularization hyperparameter α and without regularization. We indicate the cases where the computation of the condition number was unstable, as described in Section 5.3, with N/A.

Dataset	Comparison of model selection					
	valid. α			$\alpha = 0$		
	Val	Cond	Entropy	Val	Cond	Entropy
DD	0.786 ±0.006	0.787 ±0.002	0.771 ±0.009	0.763 ±0.008	0.769 ±0.005	0.750 ±0.009
ENZYMES	0.646 ±0.021	0.579 ±0.011	0.608 ±0.016	0.647 ±0.022	0.581 ±0.013	0.563 ±0.014
PROTEINS	0.741 ±0.009	0.730 ±0.004	0.744 ±0.006	0.701 ±0.011	0.713 ±0.010	0.615 ±0.013
NCI1	0.811 ±0.004	N/A	0.789 ±0.006	0.790 ±0.008	N/A	0.728 ±0.007
PTC_MR	0.626 ±0.014	N/A	0.600 ±0.018	0.614 ±0.026	N/A	0.582 ±0.035
IMDB-MULTI	0.414 ±0.024	N/A	0.408 ±0.004	0.404 ±0.008	N/A	0.398 ±0.005
IMDB-BINARY	0.669 ±0.005	N/A	0.666 ±0.012	0.664 ±0.006	N/A	0.654 ±0.007

Table 4
Comparison of test performance (accuracy, with standard deviation) of U-GCN (first column of Table 3) with results of models from the literature, as reported in the cited papers, on untrained graph networks (PyramidalRNGG [46] and FDGNN [15]), and with the best performing model for each dataset found by the comparison in [45]. The best result for each dataset is indicated in bold, and missing results are indicated with N/A.

Dataset	Model			
	U-GCN	P-RGNN	FDGNN	Best from [45]
DD	**0.786 ± 0.006**	0.737 ± 0.012	N/A	0.766 ± 0.043
ENZYMES	**0.646 ± 0.021**	0.472 ± 0.045	N/A	0.596 ± 0.045
PROTEINS	0.741 ± 0.009	0.711 ± 0.026	**0.768 ± 0.029**	0.737 ± 0.035
NCI1	**0.811 ± 0.004**	0.705 ± 0.019	0.778 ± 0.016	0.800 ± 0.014
PTC_MR	0.626 ± 0.014	N/A	**0.634 ± 0.054**	N/A
IMDB-MULTI	0.414 ± 0.024	0.497 ± 0.008	**0.500 ± 0.013**	0.485 ± 0.033
IMDB-BINARY	0.669 ± 0.005	0.723 ± 0.014	**0.724 ± 0.036**	0.712 ± 0.039

indicator of good generalization. In the absence of regularization, we find that the condition number serves as a superior performance indicator: this may be due to the fact that regularization increases the generalization ability of the models, thus the bound given with Algorithmic Stability is more indicative in its absence. As we previously observed, over-parameterization does not seem to be beneficial for the IMDB datasets, and this is reflected in the very low correlation with entropy for IMDB-MULTI, while for IMDB-BINARY the still significant correlation indicates that entropy can reflect the lower performance with increasing number of units, as seen in plot 2(d).

Moreover, in Table 3 we show that, thanks to this high correlation, in most cases the two considered measures can be used to perform model selection: selecting the hyperparameters that show better Algorithmic Stability or higher representation richness leads to test performances comparable or slightly lower than with standard grid-search model selection by using performance on the validation set. However, there are exceptions, such as the ENZYMES dataset, where the gap is more pronounced. We highlight how this kind of model selection allowed by the two metrics is the more surprising considering that it is subject to substantial limitations: it is performed without information on the task to perform, and does not require the fitting of any model except for the one with the selected hyperparameters, since only the fast feature extraction is required to compute the metrics, which are also computationally cheaper than training models. This approach could thus be used to obtain cheaply and in an unsupervised fashion (without requiring task-specific labels) relatively good model hyperparameters for untrained graph networks. The particular case of IMDB, on which we did not observe significant correlation but we see here good model selection, is due to low sensitivity of the performance to hyperparameters observed on these datasets.

Finally, we compare the results of the model we considered, U-GCN [9], with those of two other relevant randomized untrained models for graphs embedding in the literature: PyramidalRGNN [46] and FDGNN [8]. In Table 4 we show the comparison with the results reported in these two papers, and with the results of the best graph neural network model identified with the fair model comparison whose validation procedure we used [45]. While we stress how the purpose of our paper is in the analysis of the effects of over-parameterization using graph random features and in the use of average feature entropy and hypothesis stability to assess them, we observe how the untrained graph convolution is competitive with (or generally comparable to) the results in the literature, with the exception of the IMDB datasets which we identified as cases in which over-parameterization is not beneficial. Furthermore, by comparing with Table 3, we observe that for the datasets DD, ENZYMES and NCI1 on which U-GCN appears to have an advantage over the other models, even a model selection using entropy would still result in stronger results.

7. Conclusions

In this paper, we studied the generalization abilities of over-parameterized neural models based on graph random features, analyzing Algorithmic Stability and representation richness. We observed how the condition number of the Gram matrix of graph representations and average feature entropy are good indicators for the generalization of the models, and are able to identify when over-parameterization is beneficial for learning. The applicability of conditioning computations varies across datasets, presenting challenges in its reliability as an indicator of model performance. When accurately assessed (on three out of seven considered datasets), the conditioning of the Gram matrix proves to be significantly correlated with model performance in the absence of regularization. Entropy provides a consistent and easy to compute metric across datasets, exhibiting a good correlation with model accuracy, especially in scenarios where regularization is applied. Thanks to their good correlation with performance, we showed how the considered metrics can be used for model selection without requiring supervision and achieving in most cases comparable results to a classic grid search approach. Further research could be conducted to refine this unsupervised hyperparameter selection strategy to facilitate its practical application.

This work represents a step in understating over-parameterized neural models based on graph random features, yet moving forward there are still several aspect that require further study. Our analysis is conducted with a fixed number of layers and with just one model architecture: deeper models and alternative architectures could unveil new insights into their generalization capabilities. Additionally, a layer-wise analysis using the considered metrics may offer an avenue to understand how information processing varies across different layers, which could lead to the development of effective dynamic network growth strategies.

CRediT authorship contribution statement

Giovanni Donghi: Writing – original draft, Visualization, Software, Investigation. **Luca Pasa:** Writing – review & editing, Supervision, Conceptualization. **Luca Oneto:** Writing – original draft, Conceptualization. **Claudio Gallicchio:** Conceptualization. **Alessio Micheli:** Conceptualization. **Davide Anguita:** Conceptualization. **Alessandro Sperduti:** Conceptualization. **Nicolò Navarin:** Writing – review & editing, Supervision, Software, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Prof. Luca Oneto and Prof. Nicoló Navarin are associate editors of Neurocomputing. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is publicly available, code is provided.

Acknowledgments

We acknowledge the support of:

- the Italian National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No. 3277 of 30 December 2021 of Italian Ministry of University and Research funded by the European Union – NextGenerationEU:
  - project code PE0000013, Concession Decree No. 1555 of October 11, 2022, CUP C63C22000770006, project title “Future AI Research (FAIR) - Spoke 2 Integrative AI - Symbolic conditioning of Graph Generative Models (SymboliG)”;
  - project code ECS00000043, Concession Decree No. 1058 of June 23, 2022, CUP C43C22000340006, project title “iNEST: Interconnected Nord-Est Innovation Ecosystem”;
  - project code PE00000014, project title SERICS;
  - project code PE00000013, project title “FAIR - Future Artificial Intelligence Research” - Spoke 1 “Human-centered AI”, funded by the European Commission under the NextGeneration EU programme;
- the PON R&I 2014–2020 project *Smart Waste Treatment* founded by the FSE REAC-EU;
- NEURONE, a project funded by the Italian Ministry of University and Research (PRIN 20229JRTZA)
- EMERGE, a project funded by EU Horizon research and innovation programme (GA n. 101070918);
- the project ELSA – European Lighthouse on Secure and Safe AI funded by the European Union’s Horizon Europe under grant agreement No. 101070617;
- the project “Lifelong Learning on large-scale and structured data”, funded by the EC-funded OCRE project.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2024.128281.

References

[1] Z. Chen, H. Schaeffer, Conditioning of random feature matrices: double descent and generalization error, 2021, arXiv preprint arXiv:2110.11477.

[2] F. Liu, J. Suykens, V. Cevher, On the double descent of random features models trained with sgd, in: Neural Information Processing Systems, 2022.

[3] L. Oneto, S. Ridella, D. Anguita, Do we really need a new theory to understand over-parameterization? Neurocomputing 543 (2023) 126227.

[4] A. Rangamani, L. Rosasco, T. Poggio, For interpolating kernel machines, minimizing the norm of the ERM solution maximizes stability, Anal. Appl. (Singap.) 21 (01) (2023) 193–215.

[5] T. Poggio, G. Kur, A. Banburski, Double descent in the condition number, 2019, arXiv preprint a.

[6] B. Ghorbani, S. Mei, T. Misiakiewicz, A. Montanari, Linearized two-layers neural networks in high dimension, Ann. Statist. 49 (2) (2021) 1029–1054.

[7] E.H. Lee, V. Cherkassky, Understanding double descent using VC-theoretical framework, IEEE Trans. Neural Netw. Learn. Syst. (2024) 1–10.

[8] C. Gallicchio, A. Micheli, Fast and deep graph neural networks, in: AAAI Conference on Artificial Intelligence, 2020.

[9] N. Navarin, L. Pasa, C. Gallicchio, A. Sperduti, An untrained neural model for fast and accurate graph classification, in: International Conference on Artificial Neural Networks, 2023.

[10] N. Navarin, L. Pasa, L. Oneto, A. Sperduti, An empirical study of over-parameterized neural models based on graph random features, in: ESANN 2023 proceesdings, 2023, pp. 17–22, http://dx.doi.org/10.14428/esann/2023.es2023-145.

[11] C. Gallicchio, A. Micheli, Architectural richness in deep reservoir computing, Neural Comput. Appl. 35 (34) (2023) 24525–24542.

[12] M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comp. Sci. Rev. 3 (3) (2009) 127–149.

[13] H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.

[14] C. Gallicchio, A. Micheli, Graph echo state networks, in: International Joint Conference on Neural Networks, 2010.

[15] C. Gallicchio, A. Micheli, Fast and deep graph neural networks, Proc. AAAI Conf. Artif. Intell. 34 (04) (2020) 3898–3905.

[16] L. Pasa, N. Navarin, A. Sperduti, Multiresolution reservoir graph neural network, IEEE Trans. Neural Netw. Learn. Syst. 33 (6) (2022) 2642–2653.

[17] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, X. Wu, M. Li, Are graph convolutional networks with random weights feasible? IEEE Trans. Pattern Anal. Mach. Intell. 45 (3) (2023) 2751–2768.

[18] M. Loog, T. Viering, A. Mey, et al., A brief prehistory of double descent, Proc. Natl. Acad. Sci. 117 (20) (2020) 10625–10626.

[19] OpenAI, GPT-4 technical report, 2023, arXiv preprint arXiv:2303.08774.

[20] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[21] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, 2010.

[22] A. Dempster, F. Petitjean, G.I. Webb, ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (2020) 1454–1495.

[23] M.C. Ozturk, D. Xu, J.C. Príncipe, Analysis and design of echo state networks, Neural Comput. 19 (1) (2007) 111–138.

[24] T.L. Carroll, Optimizing reservoir computers for signal classification, Front. Physiol. 12 (2021) 893.

[25] J. Principe, Information theoretic learning: Renyi’s entropy and kernel perspectives, in: Information Science and Statistics, Springer New York, 2010.

[26] V.N. Vapnik, Statistical learning theory, Wiley New York, 1998.

[27] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: from theory to algorithms, Cambridge University Press, 2014.

[28] L. Oneto, Model selection and error estimation in a nutshell, Springer, 2020.

[29] O. Bousquet, A. Elisseeff, Stability and generalization, J. Mach. Learn. Res. 2 (2002) 499–526.

[30] L. Oneto, A. Ghio, S. Ridella, D. Anguita, Fully empirical and data-dependent stability-based bounds, IEEE Trans. Cybern. 45 (9) (2014) 1913–1926.

[31] A. Elisseeff, T. Evgeniou, M. Pontil, L.P. Kaelbing, Stability of randomized learning algorithms, J. Mach. Learn. Res. 6 (1) (2005) 55–79.

[32] T. Poggio, R. Rifkin, S. Mukherjee, P. Niyogi, General conditions for predictivity in learning theory, Nature 428 (6981) (2004) 419–422.

[33] S. Shalev-Shwartz, O. Shamir, N. Srebro, K. Sridharan, Learnability, stability and uniform convergence, J. Mach. Learn. Res. 11 (2010) 2635–2670.

[34] S. Mukherjee, P. Niyogi, T. Poggio, R. Rifkin, Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization, Adv. Comput. Math. 25 (1) (2006) 161–193.

[35] A. Maurer, A second-order look at stability and generalization, in: Conference on Learning Theory, 2017.

[36] L. Devroye, T. Wagner, Distribution-free inequalities for the deleted and holdout error estimates, IEEE Trans. Inform. Theory 25 (2) (1979) 202–207.

[37] M. Kearns, D. Ron, Algorithmic stability and sanity-check bounds for leave-one-out cross-validation, in: International Conference on Computational Learning Theory, 1997.

[38] B. Efron, R.J. Tibshirani, An introduction to the bootstrap, CRC Press, 1994.

[39] A. Kleiner, A. Talwalkar, P. Sarkar, M.I. Jordan, A scalable bootstrap for massive data, J. R. Stat. Soc. Ser. B Stat. Methodol. 76 (4) (2014) 795–816.

[40] C. Morris, N.M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, TUDataset: a collection of benchmark datasets for learning with graphs, in: ICML 2020 Workshop on Graph Representation Learning and beyond (GRL+ 2020), 2020.

[41] K.M. Borgwardt, C.S. Ong, S. Schönauer, et al., Protein function prediction via graph kernels, Bioinformatics 21 (2005) i47–i56.

[42] N. Wale, I.A. Watson, G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, Knowl. Inf. Syst. 14 (2008) 347–375.

[43] C. Helma, R.D. King, S. Kramer, A. Srinivasan, The predictive toxicology challenge 2000–2001, Bioinformatics 17 (1) (2001) 107–108.

[44] P. Yanardag, S. Vishwanathan, Deep graph kernels, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’15, 2015, pp. 1365–1374.

[45] F. Errica, M. Podda, D. Bacciu, A. Micheli, A fair comparison of graph neural networks for graph classification, in: Proceedings of the 8th International Conference on Learning Representations, ICLR, 2020.

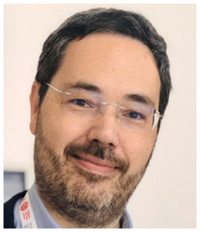
[46] F. Bianchi, C. Gallicchio, A. Micheli, Pyramidal reservoir graph neural network, Neurocomputing 470 (2022) 389–404.

Giovanni Donghi received both his B.Sc. in Mathematics and M.Sc. in Data Science with honors from the University of Padua (UniPd). During his undergraduate and graduate studies, he was a student of the Galileian School of Higher Education (SGSS), the School of Excellence at UniPd. Currently, he is a Ph.D. candidate at the Department of Mathematics, "Tullio Levi-Civita", at the same institution, in the Brain, Mind & Computer Science (BMCS) program. Working under the supervision of Professor Nicolò Navarin, his main research interests lie in the field of machine learning, in particular graph neural networks and continual learning.

Luca Pasa earned his Master's degree in Computer Science from the University of Padova in 2013. Subsequently, at the same institution, he successfully completed his Ph.D. in Mathematical Sciences (curriculum Computer Science) in March 2017, under the supervision of Professor Alessandro Sperduti. Between July 2017 and June 2019, Luca served as a postdoctoral researcher at the Center of Translational Neurophysiology Speech and Communication (CTNSC) within the Istituto Italiano di Tecnologia (IIT), working under the supervision of Dr. Leonardo Badino. Thereafter, from July 2019 to December 2021, he continued his research activity as a postdoctoral researcher at the University of Padova, affiliated with the Department of Mathematics and the Human Inspired Technologies Research Center. Since January 2022, Luca has held the position of Assistant Professor (RTDa) at the Department of Mathematics, University of Padova. His main research interests lie in the field of Machine Learning, including Deep Learning, Computational Neuroscience, and Automatic Speech Recognition. Currently, his research activity is focused on the application of Deep Learning methods on structured domains (sequences, trees, and graphs). Luca has co-authored numerous research papers published in international refereed journals and conference proceedings. He has also served as a Program Committee member for several machine learning conferences and has actively participated in organizing various special sessions at international machine learning conferences. He is an associate editor for the journal IEEE Transactions On Neural Networks And Learning Systems (TNNLS) and he is a member of the IEEE Task Force on Deep Learning and the Italian Association for Artificial Intelligence (AIxIA).

Luca Oneto was born in Rapallo, Italy in 1986. He received his B.Sc. and M.Sc. in Electronic Engineering at the University of Genoa, Italy respectively in 2008 and 2010. In 2014 he received his Ph.D. from the same university in the School of Sciences and Technologies for Knowledge and Information Retrieval with the thesis "Learning Based On Empirical Data". In 2017 he obtained the Italian National Scientific Qualification for the role of Associate Professor in Computer Engineering and in 2018 he obtained the one in Computer Science. He worked as Assistant Professor in Computer Engineering at University of Genoa from 2016 to 2019. In 2018 he was co-funder of the spin-off ZenaByte s.r.l. In 2019 he obtained the Italian National Scientific Qualification for the role of Full Professor in Computer Science and Computer Engineering. In 2019 he became Associate Professor in Computer Science at University of Pisa and currently is Associate Professor in Computer Engineering at University of Genoa. He has been involved in several H2020 projects (S2RJU, ICT, DS) and he has been awarded with the Amazon AWS Machine Learning and Somalvico (best Italian young AI researcher) Awards. His first main topic of research is the Statistical Learning Theory with particular focus on the theoretical aspects of the problems of (Semi) Supervised Model Selection and Error Estimation. His second main topic of research is Data Science with particular reference to the problem of Trustworthy AI and the solution of real world problems by exploiting and improving the most recent Learning Algorithms and Theoretical Results in the fields of Machine Learning and Data Mining.

Claudio Gallicchio is a Tenured-track Assistant Professor of Machine Learning at the Department of Computer Science of the University of Pisa, Italy. He received his Ph.D. from the University of Pisa in 2011, where he focused on Recurrent and Reservoir Computing models and theory for structured data. His research is based on the fusion of concepts from Deep Learning, Recurrent Neural Networks, Neuromorphic Computing, and Randomized Neural Systems. He is the founder and former chair of the IEEE CIS Task Force on Reservoir Computing, co-founder and vice-chair of the IEEE Task Force on Randomization-based Neural Networks and Learning Systems.



Alessio Micheli is Full Professor at the Department of Computer Science of the University of Pisa, where he is the head and scientific coordinator of the Computational Intelligence & Machine Learning Group (CIML), part of the CLAIRE-AI.org Research Network. His research interests include machine learning, neural networks, deep learning, learning in structured domains (sequence, tree, and graph data), recurrent and recursive neural networks, reservoir computing, and probabilistic and kernel-based learning for nonvectorial data, with an emphasis on efficient neural networks for learning from graphs. Prof. Micheli is the national coordinator of the "Italian Working group on Machine Learning and Data Mining" of the Italian Association for Artificial Intelligence and he has been co-founder/chair of the IEEE CIS Task Force on Reservoir Computing. He is an elected member of the Executive committee of the European Neural Network Society – ENNS. He also serves as an Associate Editor for *Neural Networks* and for *IEEE Transactions on Neural Networks and Learning Systems*.

Davide Anguita received the 'Laurea' degree in Electronic Engineering and a Ph.D. degree in Computer Science and Electronic Engineering from the University of Genoa, Genoa, Italy, in 1989 and 1993, respectively. After working as a Research Associate at the International Computer Science Institute, Berkeley, CA, on special-purpose processors for neurocomputing, he returned to the University of Genoa. He is currently Associate Professor of Computer Engineering with the Department of Informatics, BioEngineering, Robotics, and Systems Engineering (DIBRIS). His current research focuses on the theory and application of kernel methods and artificial neural networks.

Alessandro Sperduti received the Ph.D. in 1993 from University of Pisa, Italy. He is a Full Professor at the Department of Mathematics of the University of Padova. Previously, he has been associate professor (1998–2002) and assistant professor (1995–1998) at the Department of Computer Science of the University of Pisa. His research interests are mainly in Neural Networks, Kernel Methods, and Process Mining. He was the recipient of the 2000 AI*IA (Italian Association for Artificial Intelligence) "MARCO SOMALVICO" Young Researcher Award. He has been invited as plenary speaker in several Neural Networks conferences. Prof. Sperduti has served as AC in major AI Conferences and currently is in the editorial board of the journals Theoretical Computer Science (Section C), Natural Computing, Neural Networks. He has been member of the European Neural Networks Society (ENNS) Executive Committee, chair of the DMTC of IEEE CIS for the years 2009 and 2010, chair of the NNTC for the years 2011 and 2012, chair of the IEEE CIS Student Games-Based Competition Committee for the years 2013 and 2014, and Chair of the Continuous Education Committee of the IEEE Computational Intelligence Society for yer 2015. He is senior member IEEE. Prof. Sperduti is the author of more than 220 publications on refereed journals, conferences, and chapters in books.

Nicolò, Navarin is associate professor in Computer Science at the Department of Mathematics "Tullio Levi-Civita", University of Padua, Italy. He got his Ph.D. in computer science from the University of Bologna, Italy, in 2014. He has been a visiting Researcher at the University of Freiburg, Germany, at the Università della Svizzera Italiana, Lugano, Switzerland, and at 3IA Côte d'Azur - Interdisciplinary Institute for Artificial Intelligence, Sophia Antipolis, France. He has been a research fellow at the University of Nottingham, UK and at the University of Padua. His research interests lie in the field of machine learning, including kernel methods and neural networks for structured data, online and continual learning, trustworthy ML, and applications to bioinformatics, business process mining, computer vision and computational psychology. Prof. Navarin has been serving as PC member in major machine learning conferences (ICML, IJCAI, NeurIPS, ECML, AAAI, ICLR), and he has been actively involved in the organization of several special sessions (ESANN, WCCI, IJCNN) and conferences (INNS BDDL 2019, ICPM 2020, IEEESSCI 2021, IEEE WCCI 2022). He is an associate editor for the journals Evolving Systems (Springer) and Neurocomputing (Elsevier). In December 2023, he obtained the National Scientific Qualification for the role of Full Professor in Computer Engineering.