

GRAPH TOPOLOGY INFERENCE BENCHMARKS FOR MACHINE LEARNING

Carlos Lassance[◇], Vincent Gripon[◇], and Gonzalo Mateos[†]

[◇] IMT Atlantique, Lab-STICC, France, [†] Dept. of ECE, University of Rochester, USA

ABSTRACT

Graphs are nowadays ubiquitous in the fields of signal processing and machine learning. As a tool used to express relationships between objects, graphs can be deployed to various ends: (i) clustering of vertices, (ii) semi-supervised classification of vertices, (iii) supervised classification of graph signals, and (iv) denoising of graph signals. However, in many practical cases graphs are not explicitly available and must therefore be inferred from data. Validation is a challenging endeavor that naturally depends on the downstream task for which the graph is learnt. Accordingly, it has often been difficult to compare the efficacy of different algorithms. In this work, we introduce several ease-to-use and publicly released benchmarks specifically designed to reveal the relative merits and limitations of graph inference methods. We also contrast some of the most prominent techniques in the literature.

Index Terms— Graph learning, network topology inference, benchmarks, graph signal processing, machine learning.

1. INTRODUCTION

Graphs are mathematical objects that express relationships between items, referred to as vertices. As a natural representation of complex data structure, graphs are ubiquitous, in particular in the field of machine learning, where they can be used for various ends: (i) they can model the inner dependencies of observations, e.g. functional connectivity in the brain [1]; (ii) they can model the relationship between data samples, e.g. social networks and citation graphs [2]; and (iii) they can be used to directly model data, e.g. gene-expression levels collected from microarray experiments [3].

However, graphs are not always explicitly available. Many recent works have therefore considered the problem of inferring the topology (i.e. the edges) of the graph based on nodal observations [4, 5, 6]; see also [7] for a recent tutorial treatment. Inferring a graph structure can be performed in a task-agnostic manner, where only unsupervised observations are considered. In such a case, priors are used to relate observations to the sought graph structure: e.g. smoothness [5], stationarity [4], sparsity [6], and probabilistic [8] as well as

graph filtering-based [9] generative models, just to name a few. Other works consider inferring a graph for a specific task. For example in [10] the authors infer graphs for medical search. In [11, 12, 13], the authors aim at improving the accuracy of various classification tasks using inferred graphs: semi-supervised learning, visual based localization and few-shot learning.

Because it is more general, task-agnostic graph inference is of particular interest. In this context, common desiderata are to generate graphs used for visualization [5] and interpretation [14]. On the other hand, it is challenging to compare methods for which there is no ground truth. Unsurprisingly, many works rely on synthetic data to evaluate the ability of their proposed methods in unveiling the topology from the observations. While synthetic data are always useful to perform controlled scalability experiments as well as reveal the emerging statistical and computational trade-offs, this validation protocol comes with two shortcomings. First, the models used to generate synthetic data are likely to be biased in favor of the proposed methods. Second, the ability of the proposed method to handle hard real-world problems is often not demonstrated convincingly.

In order to address this problem, standardized benchmarks are required. The main challenge is that benchmarks are necessarily task-specific, and as such they do not encompass the whole potential offered by state-of-the-art methods. To fill in this gap, in this work we introduce a broad collection of benchmarks that are specifically designed to reveal the relative merits and limitations of graph inference algorithms. To this end, we consider three timely problems arising with network data: (i) unsupervised clustering of vertices; (ii) semi-supervised classification of vertices (with or without vertex features); and (iii) graph signal denoising. For each problem we introduce a balanced and easy-to-use dataset that we release publicly ¹. Note that our work is meant to benchmark the graph inference task, for a benchmark of the unsupervised/semi-supervised methods themselves we refer the reader to OGN [15]. Furthermore, the released datasets comprise various types of signals, namely natural images, audio, texts, and traffic information. Note that we do not include brain data and protein-protein interactions that are two of the most interesting use-cases of graph inference and

¹This work was supported in part by the Brittany region and by the NSF award CCF-1750428.

¹https://github.com/cadurosar/benchmark_graphinference

classification. Our choice is informed by recent developments in the literature [16, 17], that have found no significant performance gains when graph-based machine learning techniques are brought to bear for some tasks in these areas. As our objective here is to compare graph inference methods and not the techniques used in the downstream tasks, we will not delve into this issue further.

All in all, the contributions of this work can be summarized as follows. We assemble a diverse set of datasets for various tasks and types of signals, meant to assess the efficacy of graph inference methods. We compare selected prominent methods from the literature and identify their relative strengths and shortcomings across different tasks and types of data. We provide a public release of prepackaged data and a simple script to evaluate future methods, facilitating comparisons with the graph learning algorithms considered in this paper. The outline for the remainder of this paper is as follows. In Section 2 we introduce the considered tasks. In Section 3, we present the created datasets. In Section 4, we review some of the main methods from the literature. In Section 5, we perform experiments and discuss the results. Finally, we conclude in Section 6.

2. PROBLEM STATEMENT

We consider three tasks that can benefit from graph inference methods. These tasks were chosen to represent most widely considered applications cases found in the recent literature.

Before getting to the details of the methods, let us make a quick recall about graphs and graph signals. A graph $G = \langle V, \mathbf{W} \rangle$ is a tuple where V is the finite set of vertices and $\mathbf{W} \in \mathbb{R}^{|V| \times |V|}$ is the adjacency matrix. Typically, $\mathbf{W}$ is symmetric. The degree matrix of the graph is the diagonal matrix $\mathbf{D}_{\mathbf{W}}$ where $\mathbf{D}_{\mathbf{W}}[i, i] = \sum_j \mathbf{W}[i, j]$. Note that $\mathbf{W}[i, j]$ refers to the weight located at i -th row and j -th column in $\mathbf{W}$. Some authors like to consider normalized adjacency matrices, such as $\mathbf{W} \leftarrow \mathbf{D}_{\mathbf{W}}^{-1/2} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-1/2}$ or $\mathbf{W} \leftarrow \mathbf{D}_{\mathbf{W}}^{-1} \mathbf{W}$. The graph Laplacian is the matrix $\mathbf{L} = \mathbf{D}_{\mathbf{W}} - \mathbf{W}$. A graph signal is a (most of the time real-valued) matrix $\mathbf{X} \in \mathbb{R}^{|V| \times F}$, where F stands for the number of nodal features.

Because the Laplacian is symmetric and real-valued, it can be eigendecomposed as $\mathbf{L} = \mathbf{F} \mathbf{\Lambda} \mathbf{F}^{\top}$, where $\mathbf{F}$ is orthonormal and $\mathbf{F}^{\top}$ is its transpose; and $\mathbf{\Lambda}$ is diagonal, and its elements are sorted in ascending order. We refer to the first columns of $\mathbf{F}$ as the low-frequency eigenvectors of the Laplacian.

Consider a given set of observations, each one composed of several features. We divide our benchmarks into two types of machine learning tasks. In the first ones (Tasks 1 and 2), the graph model dependencies between observations. As such, a vertex in the graph corresponds to one observation, and is associated with the corresponding features. In the second one (Task 3), the graph models relationships between features.

Therefore, here a vertex in the graph represents a feature and the graph is used as a proxy to the topology of the signal. We expect some methods will perform better on the first series of tasks and others to be more adequate to the second one.

2.1. Task 1: Unsupervised Clustering of Vertices (UCV)

Consider a dataset composed of $|V| = N$ observations, each one containing F features. Given a number of classes C , we consider the task of partitioning the N observations into C classes, such that the variability inside classes is smaller than the variability between classes. In practice, variability can be measured using various metrics. For the purpose of obtaining quantified benchmarks, we consider here that the observations belong to C categories (e.g. classes of images or sounds), and that this information is not available when processing the considered methods. So, the performance of a considered method is evaluated by computing the Adjusted Mutual Information score [18] based on the ground truth.

Note that this clustering problem can be treated without a graph structure. Examples are using C -means or DB-Scan algorithms. In the context of this work, we consider using spectral clustering. Spectral clustering consists in creating a graph linking the observations where the edges are inferred from the corresponding features. Then, vertices are projected using the first eigenvectors of the graph Laplacian and clustered using standard non-graph methods. In our work, we use the discretization method first where features are projected onto the first C eigenvectors of the graph Laplacian except the very first one. We use the default SciKit-Learn implementation of spectral clustering and of the C -means algorithm in our experiments.

2.2. Task 2: Semi-Supervised Classification of Vertices (SSCV)

Consider a dataset composed of $|V| = N$ observations, each one containing F features. Here, a portion of the N observations are labeled. The task consists in inferring the labels of the other portion of observations. Again, we consider datasets where we have access to the ground truth, and artificially hide the labels of part of the observations when processing the data. The score consists in measuring the accuracy of the classification on initially unlabeled observations.

This problem can be solved without relying on graphs. For example, a common solution would consist in performing a supervised classification using only the labeled observations. In this work, we consider inferring a graph connecting observations from the features. Then, we use this graph in two settings. In the first setting, we want the graph to fully encompass the information contained in the features, and therefore perform label propagation. Label propagation consists in diffusing the labels from the known observations to the other ones using the inferred graph structure. In a second setting,

we use both the graph structure and the features to perform classification. We use the methodology described in [19], called Simplified Graph Convolution (SGC), where the goal is to combine feature diffusion with logistic regression. Note that in our case we should obtain equivalent results for both SGC and GCN, as the number of observed nodes is smaller than the amount of features as noted by [20].

In more detail, we use two layers of feature diffusion ($\hat{\mathbf{X}} = \mathbf{W}^2 \mathbf{X}$), followed by a logistic regression. The models are trained for 100 epochs, using Adam optimization with a learning rate of 0.001. We use the average over 100 runs of the accuracy using random splits of 5% training set and 95% test set. We always report the average accuracy and standard deviation. To propagate labels, we simply diffuse the label signal one time using the exponential of the adjacency matrix. We note that SGC models tend to use the “normalized augmented adjacency matrix” $\tilde{\mathbf{W}} = \mathbf{I} + \mathbf{W}$ where $\mathbf{I}$ is the identity matrix. This augmented adjacency matrix is then normalized $\tilde{\mathbf{W}} \leftarrow \mathbf{D}_{\tilde{\mathbf{W}}}^{-1/2} \tilde{\mathbf{W}} \mathbf{D}_{\tilde{\mathbf{W}}}^{-1/2}$. In our work we test both the adjacency matrix and the augmented adjacency matrix and their respective normalizations and we report the best possible combination in terms of mean accuracy.

2.3. Task 3: Denoising of Graph Signals (DGS)

Consider a dataset comprising N observations, each one consisting of $|V| = F$ features. Consider some additive noise generated according to a distribution $\mathcal{N}$. The task consists in recovering initial observations from their noisy versions. We measure performance by looking at the Mean Squared Error (MSE) between both.

Here, the graph connects features of observations. The idea is to use the graph structure to easily segregate components of the noise from components of the initial signals. In our work, we use a Simoncelli low-pass filter on the graph to perform denoising. In our experiments we use the PyGSP implementation of the Simoncelli filter, which is defined by its spectral response as follows:

$$f_l = \begin{cases} 1 & \text{if } \lambda_l \leq \frac{\tau}{2} \\ \cos\left(\frac{\pi}{2} \frac{\log(\lambda_l)}{\log(2)}\right) & \text{if } \frac{\tau}{2} < \lambda_l \leq \tau, \\ 0 & \text{if } \lambda_l > \tau \end{cases},$$

where $\tau \in [0, 1]$ is a user-defined threshold and λ_l the l -th Laplacian eigenvalue. We normalize the eigenvalues by dividing by the largest one, so that $0 \leq \lambda_l \leq 1$. We vary the parameter τ from 0 to 1 in increments of 0.025. We use the noisy signal with a SNR (Signal to Noise Ratio) of 7, from [21], and report the best SNR found for each graph construction.

3. DATASETS

For Tasks 1 and 2, we use datasets of images, audio and texts (documents). To reduce the difficulty of the tasks in the image and audio domains, we choose to use features extracted from pretrained deep neural networks. Task 3 (DGS) data comes from real life traffic information. Additional details are given in the coming paragraphs.

3.1. Image dataset

For the image dataset we use the training set portion of the “102 Category Flower Dataset” (shortened as flow-ers102) [22]. This split contains $N = 1020$ images of $C = 102$ classes of flowers (10 images per class). The features are extracted from the final pooling layer of the Inceptionv3 architecture [23], which has a size of $F = 2048$ dimensions. Note that Inceptionv3 was trained on the 2012 split of ImageNet challenge, so that the features we obtain are a case of transfer learning. This should be one of the most challenging scenarios we consider, as it provides the highest number of classes and has the highest signal dimension to number of items ratio: 2.

3.2. Audio dataset

For audio data, we use “ESC-50: Dataset for Environmental Sound Classification” [24]. This dataset contains $C = 50$ classes, with 40 audio signals each (2000 in total). It also contains 5 standard splits that are not used here (as we do unsupervised and semi-supervised classification). We use the feature extractor introduced in [25] to generate our dataset, that was trained on AudioSet. Similar to the images data, this can be considered as transfer. At the end we have $N = 2000$ items with $F = 1024$ dimensions each. The signal dimension to number of items ratio is 0.512.

3.3. Text dataset

We use the cora dataset [2], which is composed of $N = 2708$ scientific articles of $C = 7$ different domains for document clustering or classification. The features come from a word indicator vector (i.e. Bag of Words BoW) that indicates if one of the words in the dictionary ($F = 1433$ in total) is present on the title or abstract of the document. We prefer simple BoW because our first tests using features extracted from pretrained networks led to worse performance. The dictionary is built with the most common words in the dataset. The signal dimension to number of items ratio is: 0.53. Note that this dataset is classically used for graph semi-supervised learning as it comes with a citation graph. But in our work we completely disregard this graph. Comparisons between the ground truth graph and inferred ones could be an interesting addition to this work. But since the citation graph is not exactly redundant with the signals, it is expected that inferred

Table 1. Summary of the tested graph topology inference methods.

Method	Similarity/Distance	k	σ	Adjacency matrices
Naive	Cosine, Covariance, RBF	5, 10, 20, 30, 40, 50, 100, 200, 500, 1000	None	$\mathbf{W}, \mathbf{D}_{\mathbf{W}}^{-1/2} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-1/2},$
NNK [6]			10^{-4}	$\tilde{\mathbf{W}}, \mathbf{D}_{\tilde{\mathbf{W}}}^{-1/2} \tilde{\mathbf{W}} \mathbf{D}_{\tilde{\mathbf{W}}}^{-1/2}$
Kalofolias [5]	Square Euclidean distance			

graphs and citation ones are quite different.

3.4. Toronto traffic data denoising (Toronto)

We use data from the road network of the city of Toronto, from [21]. It describes traffic volume data at intersections in the road network of Toronto for a total of $F = 2202$ vertices and $N = 1$ observation. Note that extra information is available, such as the position of each road and intersection, but our baselines only consider the raw signal data.

4. GRAPH INFERENCE METHODS

In our work, we perform experiments using off-the-shelf graph inference techniques from the literature. We also provide implementations of the chosen techniques. Table 1 presents a summary of the methods and variations we tested.

4.1. Naive baselines

We first consider naive baselines by combining three steps:

1. Choosing a similarity measure to be applied to either features of each vertex for Tasks 1 and 2 or to observations for Task 3. In more details, we consider cosine similarity, sampled covariance or an RBF kernel applied on the L_2 distance between considered items.
2. Choosing a number of neighbors to be kept for each vertex. We simply use a k -nearest neighbor selection. Note that we symmetrize the resulting graph, so that each vertex has at least k neighbors.
3. Normalizing the obtained graph adjacency matrix.

Note that we obtain a large number of possible combinations, and perform experiments for each one. In Section 5 we only display the results obtained by the best combination.

4.2. Sparsity-based method

We now consider a more recent sparsity-based method. We choose NNK (Non Negative Kernel regression) [6], due to its simplicity and its demonstrated results on semi-supervised learning tasks. This method can be interpreted as producing representations with orthogonal approximation errors, which in turn favors sparser representations. It has two parameters: k , the maximum degree for each vertex, and σ the minimum value for an edge weight (threshold). In this work we test multiple values of k and fix $\sigma = 10^{-4}$ [5]. In our experiments we use the authors implementation from https://github.com/STAC-USC/PyNNK_graph_construction.

4.3. Smoothness-based method

For our smoothness based topology inference method, we rely on a state-of-the-art approach in [5]. It consists in a framework that infers the graph from an underlying set of smooth signals. As it was the case with the sparsity based method, it has two parameters: k the desired mean sparsity and σ the minimum value for an edge weight. We test the same values for these two parameters as we did for the previous method and keep the best combination. In our experiments we use the implementation from the GSP toolbox [26].

5. BASELINE RESULTS

5.1. Task 1

For Task 1: UCV, we display both the results obtained with the inferred graph structures and with a C -means baseline. The results are presented in Table 2. We can see that both naive and NNK get the most consistent results, with Kalofolias having difficulties with the cora dataset.

Table 2. Results for Task 1. Here we present the best AMI score for each inference method.

Method	Inference/Dataset	ESC-50	cora	flowers102
C -means		0.59	0.10	0.36
Spectral clustering	Naive	0.66	0.34	0.45
	NNK	0.66	0.34	0.44
	Kalofolias	0.65	0.27	0.44

5.2. Task 2

For the SSCV task, the results are presented in Table 3. We can see that using a similarity graph as support helps when compared to a simple logistic regression. Note that this is not a 100% fair comparison as the logistic regression is not able to exploit the unsupervised data. In this task we have two methods, Label Propagation and SGC. In the first one, Kalofolias presents the best results for both flowers102 and ESC-50, but still struggles with the cora dataset. In SGC both Kalofolias and NNK seem to not be able to improve that much over the naive baselines.

5.3. Task 3

For the graph signal denoising task, the results are presented in Table 4. In this scenario we are not able to use neither cosine or covariance similarity. We compare our results with the ones we would obtain using the ground truth road map graph. Our RBF baselines were able to reduce the amount of

Table 3. Results for Task 2. Here we present the best mean test accuracy and its standard deviation for each inference method.

Method	Inference/Dataset	ESC-50	cora	flowers102
Logistic Regression		52.92% $\pm$ 1.9	46.84% $\pm$ 1.6	33.51% $\pm$ 1.7
Label Propagation	Naive	59.05% $\pm$ 1.8	58.86% $\pm$ 2.9	36.73% $\pm$ 1.6
	NNK	57.44% $\pm$ 2.2	58.66% $\pm$ 2.9	33.57% $\pm$ 1.6
	Kalofolias	59.16% $\pm$ 1.8	58.60% $\pm$ 3.4	37.01% $\pm$ 1.7
SGC	Naive	60.48% $\pm$ 2.0	67.19% $\pm$ 1.5	37.73% $\pm$ 1.5
	NNK	61.38% $\pm$ 2.0	66.58% $\pm$ 1.5	36.81% $\pm$ 1.5
	Kalofolias	59.36% $\pm$ 2.0	66.28% $\pm$ 1.5	37.5% $\pm$ 1.5

noise, but not at the same level as of the real road graph. The Kalofolias smooth graph was able to achieve a better SNR than the real road graph.

Table 4. Results for Task 3. Here we present the best test accuracy for each baseline.

Best SNR	Road graph	Kalofolias	RBF NNK	RBF k -NN
	10.32	10.41	9.99	9.80

5.4. Discussion on baselines

Over all tasks we can extract some lessons on graph inference:

1. **Similarity choice:** If we have multiple non-negative realizations of the signal, cosine seems the best choice. It has competitive results on all benchmarks and it does not come with a parameter (as does RBF with γ).
2. **Choosing parameter k :** The best amount of sparsity depends not only on the dataset and task, but on the similarity that was chosen. We consider the ESC-50 dataset as an example. In the spectral clustering the best k value for the k -NN graph was 30 for cosine, 5 for RBF and 20 for covariance. We note that in the graph denoising task, the best case was to not perform k -neighbors thresholding.
3. **Normalization:** Note that only our graph denoising task does not expect a normalized graph, therefore most of our better results used normalized graphs. On the graph denoising task, normalized and non-normalized graphs had similar results.
4. **Cora dataset:** The cora dataset is challenging not only because it is not class-balanced, but also because its features are binary (a bag of words, containing 1 if the word is present in the article and 0 if not). This could be a reason for the bad performance of both NNK and Kalofolias in this dataset.
5. **Sparse graphs in semi-supervised problems:** In the semi-supervised tasks, the test accuracy standard deviation over the splits was very high. This could possibly be caused by the fact the sparse graphs we use

here have more than one connected component, meaning that sometimes there could be sections of the graph that do not have any labeled vertices. One possible future direction would be to integrate a graph sampling algorithm to the problem in order to select which vertices we should label, instead of doing so randomly.

6. **Naive Baselines vs. optimization approaches:** Over our tests there was no clear winner between simply doing a naive k -NN approach and more advanced graph topology inference techniques. Kalofolias had very good performance on the Label Propagation and Denoising tasks, while NNK was consistent in SGC and Spectral Clustering, but both were not able to consistently beat the naive baseline. On the other hand, there was a clear advantage of both Kalofolias and NNK over the naive baselines when we consider the robustness of both methods to the parameter k selection.

6. CONCLUSION

We have introduced graph inference benchmarks that allows us to test different graph topology inference methods in real downstream signal and information processing tasks. We have tested naive graph inference methods and more advanced techniques in the literature. This allowed us first to verify that improving the graph inference leads to better performance on the downstream tasks and to take away some guidelines for experimentation in this domain. We note that while we tested various baselines, a more thorough analysis of the results is needed. The benchmark is available online and should be easy to use to compare newer techniques. We hope that this allows for more advances in the field and we are eager to continue improving this tool as the field advances.

7. REFERENCES

- [1] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A graph signal processing perspective on functional brain imaging," *Proceedings of the IEEE*, May 2018.
- [2] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, 2008.

- [3] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, et al., "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, 2005.
- [4] B. Paskdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Transactions on Signal and Information Processing over Networks*, 2017.
- [5] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *International Conference on Learning Representations*, 2019.
- [6] S. Shekizhar and A. Ortega, "Graph construction from data using non negative kernel regression (NNK graphs)," *arXiv preprint arXiv:1910.09383*, 2019.
- [7] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, May 2019.
- [8] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE Journal on Selected Topics in Signal Processing*, Sept. 2017.
- [9] R. Shafipour, S. Segarra, A. Marques, and G. Mateos, "Identifying the topology of undirected networks from diffused non-stationary graph signals," *IEEE Transactions on Signal Processing*, 2018.
- [10] B. Koopman, G. Zuccon, P. Bruza, L. Sitbon, and M. Lawley, "Information retrieval as semantic inference: a graph inference model applied to medical search," *Information Retrieval Journal*, 2016.
- [11] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [12] C. Lassance, Y. Latif, R. Garg, V. Gripon, and I. Reid, "Improved visual localization via graph smoothing," *arXiv preprint arXiv:1911.02961*, 2019.
- [13] Y. Hu, V. Gripon, and S. Pateux, "Exploiting unsupervised inputs for accurate few-shot classification," *arXiv preprint arXiv:2001.09849*, 2020.
- [14] R. Anirudh, P. Bremer, R. Sridhar, and J. Thiagarajan, "Influential sample selection: A graph signal processing approach," Tech. Rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2017.
- [15] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.
- [16] T. He, R. Kong, A. J. Holmes, M. Nguyen, M. R. Sabuncu, S. B. Eickhoff, D. Bzdok, J. Feng, and B. T. Yeo, "Deep neural networks and kernel regression achieve comparable accuracies for functional connectivity prediction of behavior and demographics," *NeuroImage*.
- [17] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *International Conference on Learning Representations*, 2020.
- [18] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?," in *International Conference on Machine Learning*, 2009.
- [19] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning*, 2019.
- [20] C. Vignac, G. Ortiz-Jiménez, and P. Frossard, "On the choice of graph neural network architectures," in *ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [21] J. Irion and N. Saito, "Efficient approximation and denoising of graph signals using the multiscale basis dictionaries," *IEEE Transactions on Signal and Information Processing over Networks*, 2016.
- [22] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICCVGIP*, Dec 2008.
- [23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [24] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. 2015, ACM Press.
- [25] A. Kumar, M. Khadkevich, and C. Fügen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- [26] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSP-BOX: A toolbox for signal processing on graphs," *arXiv preprint arXiv:1408.5781*, Aug. 2014.