EVALUATION METRICS FOR GRAPH GENERATIVE MODELS: PROBLEMS, PITFALLS, AND PRACTICAL SOLUTIONS

Leslie O'Bray[1,2,†], **Max Horn**[1,2,†], **Bastian Rieck**[1,2,3,*] **and Karsten Borgwardt**[1,2,*]
[1]Department of Biosystems Science and Engineering, ETH Zürich, Switzerland
[2]SIB Swiss Institute of Bioinformatics, Switzerland
[3]Institute of AI for Health, Helmholtz Munich, Germany
[†] These authors contributed equally. [*] These authors jointly supervised this work.

ABSTRACT

Graph generative models are a highly active branch of machine learning. Given the steady development of new models of ever-increasing complexity, it is necessary to provide a principled way to *evaluate* and *compare* them. In this paper, we enumerate the desirable criteria for such a comparison metric and provide an overview of the status quo of graph generative model comparison in use today, which predominantly relies on maximum mean discrepancy (MMD). We perform a systematic evaluation of MMD in the context of graph generative model comparison, highlighting some of the challenges and pitfalls researchers inadvertently may encounter. After conducting a thorough analysis of the behaviour of MMD on synthetically-generated perturbed graphs as well as on recently-proposed graph generative models, we are able to provide a suitable procedure to mitigate these challenges and pitfalls. We aggregate our findings into a list of practical recommendations for researchers to use when evaluating graph generative models.

1 INTRODUCTION

Graph generative models have become an active research branch, making it possible to generalise structural patterns inherent to graphs from certain domains—such as chemoinformatics—and actively synthesise *new* graphs (Liao et al., 2019). Next to the development of improved models, their *evaluation* is crucial. This is a well-studied issue in other domains, leading to metrics such as the 'Fréchet Inception Distance' (Heusel et al., 2017) for comparing image-based generative models. Graphs, however, pose their own challenges, foremost among them being that an evaluation based on visualisations, i.e. on *perceived* differences, is often not possible. In addition, virtually all relevant structural properties of graphs exhibit spatial invariances, e.g., connected components and cycles, are invariant with respect to rotations—that have to be taken into account by a comparison metric.

While the community has largely gravitated towards a single comparison metric, the maximum mean discrepancy (MMD) (Chen et al., 2021; Dai et al., 2020; Goyal et al., 2020; Liao et al., 2019; Mi et al., 2021; Niu et al., 2020; Podda & Bacciu, 2021; You et al., 2018; Zhang et al., 2021), neither its expressive power nor its other properties have been systematically investigated in the context of graph generative model comparison. The goal of this paper is to provide such an investigation, starting from first principles by describing the desired properties of such a comparison metric, providing an overview of what is done in practice today, and systemically assessing MMD's behaviour using recent graph generative models. We highlight some of the caveats and shortcomings of the existing status quo, and provide researchers practical recommendations to address these issues. We note here that our investigations focus on assessing the structural similarity of graphs, which is a necessary first step before one can jointly assess graphs based on structural and attribute similarity. This paper purposefully refrains from developing its *own* graph generative model to avoid any bias in the comparison.

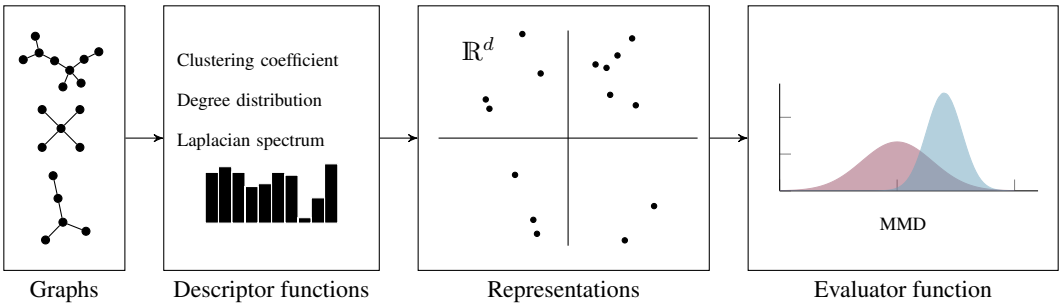


Figure 1: An overview of the workflow used to evaluate graph generative models, as is used, e.g., in Liao et al. (2019); Niu et al. (2020); You et al. (2018): given a distribution of graphs, a set of descriptor functions is employed to map each graph to a high-dimensional representation in $\mathbb{R}^d$. These representations are then compared (with a reference distribution or with each other) using an evaluator function called maximum mean discrepancy (MMD). In principle, MMD does not require the vectorial representation, but we find that this is the predominant use in the context of graph generative model evaluation.

2 Comparing graph distributions

In the following, we will deal with undirected graphs. We denote such a graph as $\mathrm{G} = (V, E)$ with vertices V and edges E. We treat graph generative models as black-box models, each of which results in a set[1] of graphs $\mathcal{G}$. The original distribution of graphs is denoted as $\mathcal{G}^*$. Given models $\{\mathcal{M}_1, \mathcal{M}_2, \dots\}$, with generated sets of graphs $\{\mathcal{G}_1, \mathcal{G}_2, \dots\}$, the goal of generative model evaluation is to assess which model is a better fit, i.e. which distribution is closer to $\mathcal{G}^*$. This requires the use of a (pseudo-)metric $\mathrm{d}(\cdot, \cdot)$ to assess the dissimilarity between $\mathcal{G}^*$ and generated graphs. We argue that the *desiderata* of any such comparison metric are as follows:

1. **Expressivity**: if $\mathcal{G}$ and $\mathcal{G}'$ do not arise from the same distribution, a suitable metric should be able to detect this. Specifically, $\mathrm{d}(\mathcal{G}, \mathcal{G}')$ should be monotonically increasing the further apart $\mathcal{G}$ and $\mathcal{G}'$ are from one another.
2. **Robustness**: if a distribution $\mathcal{G}$ is subject to a perturbation, a suitable metric should be robust to small perturbations. Changes in the metric should be ideally upper-bounded by a function of the amplitude of the perturbation. Robust metrics are preferable because of the inherent stochasticity of training generative models.
3. **Efficiency**: model comparison metrics should be reasonably fast to calculate; even though model evaluation is a *post hoc* analysis, a metric should scale well with an increasing number of graphs and an increasing size of said graphs.

While there are many ways to compare two distributions, ranging from statistical divergence measures to proper metrics in the mathematical sense, the comparison of distributions of graphs is exacerbated by the fact that individual graphs typically differ in their cardinalities and are only described up to permutation. With distances such as the graph edit distance being NP-hard in general (Zeng et al., 2009), which precludes using them as an *efficient* metric, a potential alternative is provided by using descriptor functions. A descriptor function f maps a graph G to an auxiliary representation in some space $\mathcal{Z}$. The problem of comparing a generated distribution $\mathcal{G} = \{\mathrm{G}_1, \dots, \mathrm{G}_n\}$ to the original distribution $\mathcal{G}^* = \{\mathrm{G}_1^*, \dots, \mathrm{G}_m^*\}$ thus boils down to comparing the *images* $f(\mathcal{G}) := \{f(\mathrm{G}_1), \dots, f(\mathrm{G}_n)\} \subseteq \mathcal{Z}$ and $f(\mathcal{G}^*) \subseteq \mathcal{Z}$ by any preferred statistical distance in $\mathcal{Z}$ (see Figure 1).

3 Current state of graph generative model evaluation: MMD

Of particular interest in previous literature is the case of $\mathcal{Z} = \mathbb{R}^d$ and using the *maximum mean discrepancy* $\mathrm{d}_{\mathrm{MMD}}(\cdot, \cdot)$ as a metric. MMD is one of the most versatile and expressive options available for comparing distributions of structured objects such as graphs (Borgwardt et al., 2006;

[1] Formally, the output can also be a multiset, because graphs are allowed to be duplicated.

Table 1: The kernels & parameters chosen by three graph generative models for the MMD calculation.

Model		**Kernel**	**Parameter choice σ and n_{bin}** Degree	Clustering	Laplacian
Model A	EMD	$\exp\left(W(x,y)/2\sigma^2\right)$	$\sigma = 1$, $n_{\text{bin}} = \text{maxdegree}$	$\sigma = 0.1$, $n_{\text{bin}} = 100$	N/A
Model B	TV	$\exp\left(-\frac{\mathrm{d_{TV}}(x,y)^2}{2\sigma^2}\right)$	$\sigma = 1$, $n_{\text{bin}} = \text{maxdegree}$	$\sigma = 0.1$, $n_{\text{bin}} = 100$	$\sigma = 1$, $n_{\text{bin}} = 200$
Model C	RBF	$\exp(-\lVert x-y\rVert^2/2\sigma^2)$	$\sigma = 1$, $n_{\text{bin}} = \text{maxdegree}$	$\sigma = 0.1$, $n_{\text{bin}} = 100$	$\sigma = 1$, $n_{\text{bin}} = 200$

Gretton et al., 2007), providing also a principled way to perform two-sample tests (Bounliphone et al., 2016; Gretton et al., 2012a; Lloyd & Ghahramani, 2015). It enables the comparison of two statistical distributions by means of *kernels*, i.e. similarity measures for structured objects. Letting $\mathcal{X}$ refer to a non-empty set, a function $\mathrm{k}\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if $\mathrm{k}(x_i, x_j) = \mathrm{k}(x_j, x_i)$ for $x_i, x_j \in \mathcal{X}$ and $\sum_{i,j} c_i c_j \, \mathrm{k}(x_i, x_j) \geq 0$ for $x_i, x_j \in \mathcal{X}$ and $c_i, c_j \in \mathbb{R}$. MMD uses such a kernel function to assess the distance between two distributions. Given n samples $X = \{x_1, \dots, x_n\} \subseteq \mathcal{X}$ and m samples $Y = \{y_1, \dots, y_m\} \subseteq \mathcal{X}$, the empirical estimate of the MMD between X and Y is obtained as

$$\mathrm{MMD}(X, Y) := \frac{1}{n^2} \sum_{i,j=1}^{n} \mathrm{k}(x_i, x_j) + \frac{1}{m^2} \sum_{i,j=1}^{m} \mathrm{k}(y_i, y_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{k}(x_i, y_j). \tag{1}$$

Since MMD is known to be a metric on the space of probability distributions under certain conditions, Eq. 1 is often treated as a metric as well (Liao et al., 2019; Niu et al., 2020; You et al., 2018), even though it would be more correct to only use it for hypothesis/two-sample testing, since as the authors of MMD themselves note, the empirical estimate of MMD can even take negative values (Gretton et al., 2012a), which we also observed in our experiments.[2] Nevertheless, MMD has been adopted by the community and requires (i) choosing a descriptor function f as described above, and (ii) choosing a kernel on $\mathbb{R}^d$ such as an RBF kernel. One then evaluates $\mathrm{d_{MMD}}(\mathcal{G}, \mathcal{G}^*) := \mathrm{MMD}(f(\mathcal{G}), f(\mathcal{G}^*))$ for a distribution $\mathcal{G}$. Given multiple distributions $\{\mathcal{G}_1, \mathcal{G}_2, \dots\}$, the values $\mathrm{d_{MMD}}(\mathcal{G}_i, \mathcal{G}^*)$ can be used to *rank* models: smaller values are assumed to indicate a larger agreement with the original distribution $\mathcal{G}^*$. We will now describe this procedure in more detail and highlight some of its pitfalls.

3.1 Kernels & Descriptor functions

Before calculating the MMD distance between two distributions, we need to define both the kernel function k and the descriptor function f that will convert a graph G to a representation in $\mathcal{Z} = \mathbb{R}^d$, for use in the MMD calculation in Eq. 1. We observed a variety of kernel choices in the existing literature. In fact, in three of the most popular graph generative models in use today, which we explore in detail in this paper, a different kernel was chosen for each one. These include a kernel using the first Wasserstein distance (EMD), total variation distance (TV), and the radial basis function kernel (RBF), and are listed in Table 1. In the current use of MMD, a descriptor function f is used to create a vectorial representation of a graph for use in the kernel computation. We find that several descriptor functions are commonly employed, either based on summary statistics of a graph, such as degree distribution histogram and clustering coefficient histogram, or based on spectral properties of the graph, such as the Laplacian spectrum histogram. While several papers also consider the orbit as a descriptor function, we do not consider it in depth here due to its computational complexity, which violates the "efficiency" property from our desiderata. We will now provide brief explanations of these prominent descriptor functions.

Degree distribution histogram. Given a graph $\mathrm{G} = (V, E)$, we obtain a histogram by evaluating $\deg(v)$ for $v \in V$, where position i of the resulting histogram is the number of vertices with degree i. Assuming a maximum degree d and extending the histogram with zeros whenever necessary, we obtain a mapping $f\colon \mathcal{G} \to \mathbb{R}^d$. This representation has the advantage of being easy to calculate and easy to compare; by normalising it (so that it sums to 1), we obtain a size-invariant descriptor.

[2] We will follow this convention in this paper and refer to Eq. 1 as a distance.

Clustering coefficient. The (local) clustering coefficient of a vertex v is defined as the fraction of edges within its neighbourhood divided by the number of all possible edges between neighbours, i.e.

$$C(v) := \frac{2|\{(v_i, v_j) \in E \mid v_i \text{ or } v_j \in \mathcal{N}(v)\}|}{\deg(v)(\deg(v) - 1)}. \tag{2}$$

The value of $C(v) \in [0, 1]$ measures to what extent a vertex v forms a clique (Watts & Strogatz, 1998). The collection of all clustering coefficients of a graph can be binned and converted into a histogram in order to obtain a graph-level descriptor. This function is also easy to calculate but is inherently local; a graph consisting of disconnected cliques or a fully-connected graph cannot be distinguished, for example.

Laplacian spectrum histogram. Spectral methods involve assigning a matrix to a graph G, whose spectrum, i.e. its eigenvalues and eigenvectors, is subsequently used as a characterisation of G. Let $\mathbf{A}$ refer to the *adjacency matrix* of G, with $\mathbf{A}_{ij} = 1$ if and only if vertices v_i and v_j are connected by an edge in G (since G is undirected, $\mathbf{A}$ is symmetric). The *normalised graph Laplacian* is defined as $\mathcal{L} := \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ denotes the identity matrix and $\mathbf{D}$ refers to the *degree matrix*, i.e. $\mathbf{D}_{ii} = \deg(v_i)$ for a vertex v_i and $\mathbf{D}_{ij} = 0$ for $i \neq j$. The matrix $\mathcal{L}$ is real-valued and symmetric, so it is diagonalisable with a full set of eigenvalues and eigenvectors. Letting $\lambda_1 \leq \lambda_2 \leq \ldots$ refer to the eigenvalues of $\mathcal{L}$, we have $0 \leq \lambda_i \leq 2$ (Chung, 1997, Chapter 1, Lemma 1.7). This boundedness lends itself naturally to a histogram representation (regardless of the size of G), making it possible to bin the eigenvalues and use the resulting histogram as a simplified descriptor of a graph. The expressivity of such a representation is not clear a priori; the question of whether graphs are fully determined by their spectrum is still open (van Dam & Haemers, 2003) and has only been partially answered *in the negative* for certain classes of graphs (Schwenk, 1973).

4 Issues with the current practice

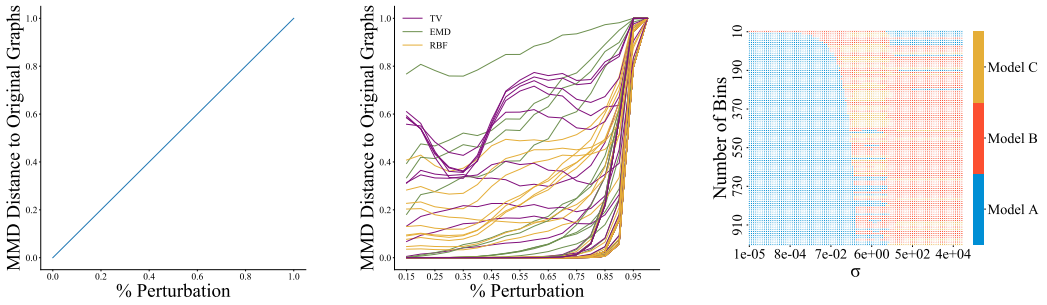
Common practice in graph generative model papers is to use MMD by fixing a kernel and parameter values for the descriptor functions and kernel values, and then assessing the newly-proposed model as well as its competitor models using said kernel and parameters. Authors chose different values of σ for the different descriptor functions, but to the best of our knowledge, all parameters are set to a fixed value a priori *without* any selection procedure. If MMD were able to give results and rank different models in a stable way across different kernel and parameter choices, this would be inconsequential. However, as our experiments will demonstrate, the results of MMD are *highly sensitive* to such choices and can therefore lead to an arbitrary ranking of models.

Subsequently, we use three current real-world models, GraphRNN (You et al., 2018), GRAN (Liao et al., 2019), and Graph Score Matching (Niu et al., 2020). We ran the models using the author-provided implementations to generate new graphs on the Community, Barabási-Albert, Erdös-Rényi, and Watts-Strogatz graph datasets, and then calculated the MMD distance between the generated graphs and the test graphs, using (i) different kernels that they used (EMD, TV, RBF), (ii) descriptor functions (degree histogram, clustering coefficient histogram, and Laplacian spectrum), and (iii) parameter ranges ($\sigma, \lambda \in \{10^{-5}, \ldots, 10^5\}$). For simplicity, we will refer to the parameter in all kernels as σ. We purposefully *refrained from using model names*, preferring instead 'A, B, C' in order to focus on the issues imposed by such an evaluation, rather than providing a commentary on the performance of a specific model.

In the following, we will delve deeper into issues originating from the individual components of the graph generative model comparison in use today. Due to space constraints, examples are provided on individual datasets; full results across datasets are available in Appendix A.6.

4.1 Nuances of using MMD for graph generative model evaluation

While MMD may seem like a reasonable first choice as a metric for comparing distributions of graphs, it is worth mentioning two peculiarities of such a choice and how authors are currently applying it in this context. First, as mentioned before, MMD was originally developed as an approach to perform two-sample testing on structured objects such as graphs. As such, its suitability was investigated in that context, and *not* in that of evaluating graph generative models. This warrants an investigation of the implications of "porting" such a method from one context to another.



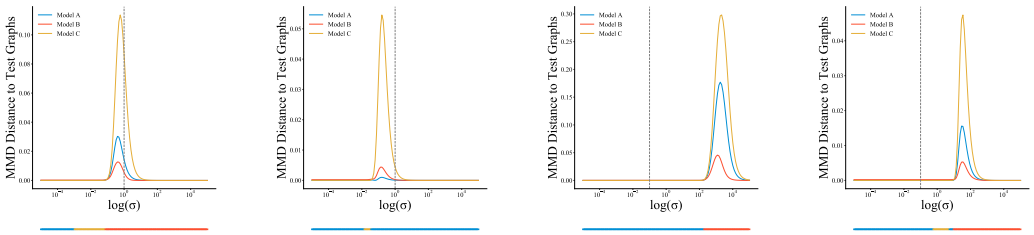
(a) (b) (c)

Figure 2: Figure 2a shows the ideal behaviour of a graph generative model evaluator: as two distributions of graphs become increasingly dissimilar, e.g. via perturbations, the metric should grow proportionally. Figure 2b shows the behaviour of the current choices in reality; each line represents the normalized MMD for a given kernel and parameter combination. A cautious choice of kernel and parameters is needed in order to obtain a metric with behaviour similar to Figure 2a. Each square in Figure 2c shows which model performs best (out of A, B, and C) over a grid of hyperparameter combinations of σ and number of bins in the histogram. Any model can rank first with an appropriate hyperparameter selection, showcasing the sensitivity of MMD to the hyperparameter choice.

The second peculiarity worth mentioning is that MMD was groundbreaking for its ability to compare distributions of structured objects by means of a kernel, thus bypassing the need to employ an intermediate vector representation. Yet, in its current application, the graphs are first being vectorised, and then MMD is used to compare the vectors. MMD is not technically required in this case: any statistical method for comparing vector distributions could be employed, such as Optimal Transport (OT). While the use of different evaluators besides MMD for assessing graph generative models is an interesting area for future research, we focus specifically on MMD, since this is what is in use today, and now highlight two practical issues that can arise from using MMD in this context.

MMD does not always capture differences in distributions. As two distributions become sufficiently dissimilar, we find that their distance should monotonically increase as a function of their dissimilarity. While one can construct specific scenarios in which two distributions become farther apart but the distance does not monotonically increase (e.g., removing edges from a triangle-free graph, and using the clustering coefficient as the descriptor function), in such cases, the specific choice of descriptor function f is crucial to ensure that f can capture differences in distribution. However, it is not guaranteed that MMD will monotonically increase as two distributions become increasingly dissimilar. Figure 2b depicts this behaviour when focusing on a single descriptor function, the clustering coefficient, with each line representing a unique kernel and parameter choice (see Appendix A.6 for the full results for other datasets and descriptor functions). We subject an original set of graphs to perturbations of increasing magnitude and then measured the MMD distance to the original distribution. Despite both distributions becoming progressively dissimilar by experimental design, a large number of kernel/parameter configurations *fail* to capture this, showing that MMD is highly sensitive to this choice. In many instances, the distance remains nearly constant, despite the increased level of perturbation, until the magnitude of the perturbation reaches an extraordinarily high level. In some cases, we observe that the distance even *decreases* as the degree of perturbation *increases*, suggesting that the original data set and its perturbed variant are more similar. We also find that the MMD values as a function of the degree of perturbation are highly sensitive to the kernel and parameter selection, as evidenced by the wide range of different curve shapes observed in Figure 2b.

MMD has no inherent scale. Another challenge with MMD is that since there is no inherent scale of MMD values, it is difficult to assess whether the smaller MMD distance of one model is *substantially* improved when compared to the MMD distance of another model. For instance, suppose that the MMD distance of one model is 5.2×10^{-8}. Is this a meaningfully better model than one whose MMD distance is 4.6×10^{-7}? This is further compounded by the choice of kernel and parameter, which also affects the scale of MMD. Since authors often choose a suboptimal value of



(a) EMD, degree (b) RBF, degree (c) EMD, clustering coef. (d) RBF, clustering coef.

Figure 3: This shows the MMD distance to the test set of graphs for three recent graph generative models (whose names we intentionally omitted) on the Community Graphs dataset for different descriptor functions and kernels. MMD requires the choice of a kernel and kernel parameters. Each subfigure shows MMD (lower is better) along a range of values of σ, with the bar underneath indicating which model ranks first for the given value of σ. The grey line indicates the σ chosen by the authors. Subfigures 3a and 3b show how simply switching from the EMD to the RBF kernel (holding σ constant) can change which model performs best; Subfigures 3c and 3d show how the choice of σ by the authors misses the area of maximum discrimination of MMD.

σ for MMD (Sutherland et al., 2017), they may obtain an arbitrarily low value of MMD, as seen in Figure 3c and 3d. As MMD results are typically reported just in a table, the lack of a scale hinders the reader's ability to effectively assess a model's performance.

4.2 Consequences of the choice of kernel

MMD requires the choice of a kernel function, yet there is no agreed-upon method to choose one. We find that each of the three models considered in this paper choose a different kernel for its evaluation, as evidenced in Table 1. There are several potential issues with the current practice, which we will subsequently discuss, namely (i) the selected kernel might be computationally inefficient to compute, as is the case for the previously-used Earth Mover's Distance (EMD) kernel function, (ii) the use of positive definite kernel functions is a limitation, with previous work employing functions that are *not* valid kernels, and (iii) finally, the kernel choice may result in arbitrary rankings, and there is currently little intention paid to this choice.

Computational cost of kernel computation. Issue (i) might prevent an evaluation metric to be used in practice, thus stymieing graph generative model development. While the choice of a kernel using the first Wasserstein distance is a valid choice, it is extremely slow to compute, as noted by Liao et al. (2019). From this perspective, it violates the third quality of our desiderata: efficiency.

Kernels need to be p.s.d. To reduce the aforementioned computational costs, previous work (Liao et al., 2019) used a kernel based on the total variation distance between histograms, which has since been used in subsequent publications as one of the ways to evaluate different models. As we show in Appendix A.1, this approach leads to an *indefinite kernel* (i.e. the kernel is neither positive definite nor negative definite), whose behaviour in the context of MMD is not well-defined. MMD necessitates the use of p.s.d. kernels, and care must be taken when it comes to interpreting the respective results.

Arbitrary ranking purely based on kernel choice. Issue (iii) relates to the fact that changing the kernel can lead to different results of model evaluation, which is problematic since each paper we considered used a different kernel. For instance, with the degree distribution as the graph descriptor function, simply changing the choice of kernel from EMD to RBF (while holding the parameters constant) leads to a different ranking of the models, which can be seen in Figure 3a–Figure 3b, where the best performing model changes merely by changing the kernel choice (!). This type of behaviour is highly undesired and problematic, as it implies that model performance (in terms of the evaluation) can be improved by choosing a different kernel.

4.3 EFFECT OF THE CHOICE OF HYPERPARAMETERS

While the choice of which kernel to use in the MMD calculation is itself an *a priori* design choice without clear justification, it is further exacerbated by the fact that many kernels require picking parameters, without any clear process by which to choose them. To the best of our knowledge, this selection of parameters is glossed over in publications at present.[3] For instance, Table 1 shows how authors are setting the value of σ differently for different descriptor functions, yet there is no discussion nor established best practice of *how* such parameters were or should be set. Hyperparameter selection is known to be a crucial issue in machine learning that mandates clear selection algorithms in order to avoid biasing the results. If the choice of parameters—similar to the choice of kernel—had no bearing on the outcome, this would not be problematic, but our empirical experiments prove that drastic differences in model evaluation performance can occur.

Arbitrary ranking based on parameter choice. The small colour bars underneath each plot of Figure 3 show the model that achieves the lowest MMD for a given value of σ. Changes in colour highlight a sensitivity to specific parameter values. Even though the plots showing MMD seem to have a general trend of which model is best in the peak of the curves, in the other regions of σ, the ranking switches, with the effect that a different model would be declared "best." This is particularly the case in Subfigures 3a, 3c, and 3d, where Model B appears to be the best, yet for much of σ, a different model ranks first. This sensitivity is further exacerbated by the fact that some descriptor functions require a parameter choice as well, such as the bin size, n_{bin} for a histogram. Figure 2c shows how the best-ranking model on the Barabási-Albert Graphs is entirely dependent upon the choice of parameters (n_{bin}, σ). The colour in each grid cell corresponds to the best-ranking model for a given σ and n_{bin}; we find that this is wildly unstable across both n_{bin} and σ. The consequence of this is alarming: *any* model could rank first if the right parameters are chosen.

Choice of σ by authors does *not* align with maximum discrimination in MMD. Figures 3c and 3d show MMD for different values of σ with the clustering coefficient descriptor function in the Community Graphs dataset for Models A, B and C. The value of σ as selected by the authors (without justification or discussion) is indicated by the grey line; however, this choice corresponds to a regions of low activity in the MMD curve, suggesting a poor parameter choice (also shown in Sutherland et al. (2017)). While Model B seems to be the clear winner, the choice of σ by the authors resulted in Model A having the best performance. Furthermore, it is not clear that choosing the same σ across different kernels, as is currently done, makes sense; whereas $\sigma = 10^3$ would be sensible for EMD in Figure 3c, for the Gaussian kernel in Figure 3d, such a choice too far beyond the discriminative peak.

5 HOW TO USE MMD FOR GRAPH GENERATIVE MODEL EVALUATION

Having understood the potential pitfalls of using MMD, we now turn to suggestions on how to better leverage MMD for graph generative model evaluation.

Provide a sense of scale. As mentioned in Section 4.1, MMD does not have an inherent scale, making it difficult to assess what is 'good.' To endow their results with some meaning, practitioners should calculate MMD between the test and training graphs, and then include this in the results table/figures alongside the other MMD results. This will provide a meaningful bound on what two 'indistinguishable' sets of graphs look like in a given dataset (see Figures 10–13 in Appendix A.6).

Choose valid and efficient kernel candidates. We strongly recommend to avoid the EMD-based kernel due to the computational burden (see Appendix A.8), and the total variation kernel for its non-p.s.d nature. Instead, we suggest using either an RBF kernel, since it is a universal kernel, or a Laplacian kernel, or a linear kernel, i.e. the canonical inner product on $\mathbb{R}^d$, since it is parameter-free. As all of these kernels are p.s.d.,[4] and fast to compute, they satisfy the efficiency desiderata criteria, and thus only require analysis of their expressivity and robustness.

[3] We remark that literature outside the graph generative modelling domain describes such choices, for example in the context of two-sample tests (Gretton et al., 2012b) or general model criticism (Sutherland et al., 2017). We will subsequently discuss to what extent the suggested parameter selection strategies may be transferred.

[4] In the case of the Laplacian kernel, the TV distance in lieu of the Euclidean distance leads to a valid kernel.

Utilize meaningful descriptor functions. Different descriptor functions measure different aspects of the graph and are often domain-specific. As a general recommendation, we propose using previously-described (Liao et al., 2019; You et al., 2018) graph-level descriptor functions, namely (i) the degree distribution, (ii) the clustering coefficient, and (iii) the Laplacian spectrum histograms, and recommend that the practitioner make domain-specific adjustments based on what is appropriate.

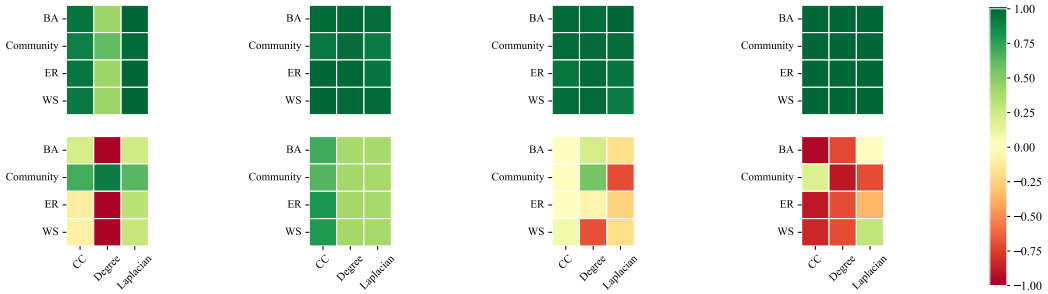
5.1 Selecting an appropriate kernel and hyperparameters

The kernel choice and descriptor functions require hyperparameter selection. We recommend assessing the performance of MMD in a controlled setting to elucidate some of the properties specific to the dataset and the descriptor functions of interest to a given application. In doing so, it becomes possible to choose a kernel and parameter combination that will yield informative results. Notice that in contrast to images, where visualisation provides a meaningful evaluation of whether two images are similar or not, graphs cannot be assessed in this manner. It is thus necessary to have a principled approach where the degree of difference between two distributions can be controlled. We subject a set of graphs to perturbations (edge insertions, removals, etc.) of increasing magnitude, thus enabling us to assess the expected degree of difference to the original graphs.

We ideally want an evaluation metric to effectively reflect the degree of perturbations. Hence, with an increasing degree of perturbation of graphs, the distance to the original distribution $\mathcal{G}^*$ of unperturbed graphs should increase. We can therefore assess both the expressivity of the evaluation metric, i.e., its ability to distinguish two distributions when they are different, and its robustness (or stability) based on how rapidly such a metric changes when subject to small perturbations. Succinctly, we would like to see a clear correlation of the metric with the degree of perturbation, thus indicating both robustness and expressivity. Perturbation experiments are particularly appealing because they do not require access to other models but rather only an initial distribution $\mathcal{G}^*$. This procedure therefore does not leak any information from models and is unbiased. Moreover, researchers have more control over the ground truth in this scenario, as they can adjust the desired degree of dissimilarity. While there are many perturbations that we will consider (adding edges, removing edges, rewiring edges, and adding connected nodes), we focus primarily on progressively adding or removing edges. This is the graph analogue to adding "salt-and-pepper" noise to an image, and in its most extreme form (100% perturbation) corresponds to a fully-connected graph and fully-disconnected graph, respectively.

Creating dissimilarity via perturbations. For each perturbation type, i.e., (i) random edge insertions, (ii) random edge deletions, (iii) random rewiring operations ('swapping' edges), (iv) random node additions, we progressively perturb the set of graphs, using the relevant perturbation parameters, in order to obtain multiple sets of graphs that are increasingly dissimilar from the original set of graphs. Each perturbation is parametrised by at least one parameter. When removing edges, for instance, the parameter is the probability of removing an edge in the graph. Thus for a graph with 100 edges and $p_{\text{remove}} = 0.1$ we would expect on average 90 edges to remain in the graph. Similar parametrisations apply for the other perturbations, i.e. the probability of adding an edge for edge insertions, the probability of rewiring an edge for edge rewiring, and the number of nodes to add to a graph, as well as the probability of an edge between the new node and the other nodes in the graph for adding connected nodes. We provide a formal description of each process in Appendix A.2 and A.3.

Correlation analysis to choose a kernel and hyperparameters. For each of the aforementioned perturbation types, we compared the graph distribution of the perturbed graphs with the original graphs using the MMD. We repeated this for different scenarios, comprising different kernels, different descriptor functions, and where applicable, parameters. For a speed up trick to efficiently calculate MMD over a range of σ, please see Appendix A.5. Since these experiments resulted in hundreds of configurations, due to the choice of kernel, descriptor function, and parameter choices, we relegated most of the visualisations to the Appendix (see Section A.6). To compare the different configurations effectively, we needed a way to condense the multitude of results into a more interpretable and comparable visualisation. We therefore calculated Pearson's correlation coefficient between the degree of perturbation and the resulting MMD distance, obtaining two heatmaps. The first one shows the *best parameter choice*, the second one shows the *worst parameter choice*, both measured in terms of Pearson's correlation coefficient (Figure 4). In the absence of an agreed-upon procedure to choose such parameters, the heatmaps effectively depict the extremes of what will happen if one is particularly "lucky" or "unlucky" in the choice of parameters. A robust combination of descriptor function and



(a) AddEdges (b) RemoveEdges (c) RewireEdges (d) AddConnectedNodes

Figure 4: The correlation of MMD with the degree of perturbation in the graph, assessed for different descriptor functions and datasets (BA: Barabási-Albert, ER: Erdös-Rényi, WS: Watts-Strogatz). For an ideal metric, the distance would increase with the degree of perturbation, resulting in values ≈ 1. The upper row shows the *best* kernel-parameter combination; the bottom row shows the *worst*. A proper kernel and parameter selection leads to strong correlation to the perturbation, but a bad choice can lead to inverse correlation, highlighting the importance of a good kernel/parameter combination.

comparison function is characterised by both heatmaps exhibiting high correlation with the degree of perturbation. In the bottom chart, the MMD distance in many cases no longer shows *any correlation* with the degree of perturbation, and in the case of the clustering coefficient (CC), is even negatively correlated with the degree of perturbation. Such behaviour is undesired, showcasing a potential pitfall for authors if they inadvertently fail to pick a "good" parameter or kernel combination. While we chose the Pearson correlation coefficient for its simplicity and interpretability, other measures of dependence could be used instead if they are domain-appropriate (see Appendix A.7). To choose a kernel and parameter combination, we suggest picking the parameters with the highest correlation for the perturbation that is most meaningful in the given domain. Lacking this, the practitioner could choose the combination that has the highest *average* correlation across perturbations.

6 Conclusion

We provided a thorough analysis of how graph generative models are being currently assessed by means of MMD. While MMD itself is powerful and expressive, its use has certain idiosyncratic issues that need to be avoided in order to obtain fair and reproducible comparisons. We highlighted some of these issues, most critical of which are that the choice of kernel and parameters can result in different rankings of different models, and that MMD may not monotonically increase as two graph distributions become increasingly dissimilar. As a mitigation strategy, we propose running a perturbation experiment as described in this paper to select a kernel and parameter combination that is highly correlated with the degree of perturbation. This way, the choice of parameters does not depend on the candidate models but only on the initial distribution of graphs.

Future work. This work gives an overview of the current situation, illuminates some issues with the status quo, and provides practical solutions. We hope that this will serve as a starting point for the community to further develop methods to assess graph generative models. Future work could investigate the use of efficient graph kernels in combination with MMD, as described in a recent review (Borgwardt et al., 2020). This would reduce the comparison pipeline in that graph kernels can be directly used with MMD, making graph descriptor functions unnecessary. Another approach could be to investigate alternative or new descriptor functions, such as the geodesic distance, or alternative evaluation methods, such as the multivariate Kolmogorov–Smirnov test (Justel et al., 1997), or even develop totally novel evaluation strategies, preferably those that go beyond the currently-employed vectorial representations of graphs.

REPRODUCIBILITY STATEMENT

We have provided the code for our experiments in order to make our work fully reproducible. Details can be found in Appendix A.2-A.4, which includes a link to our GitHub repository. During the review process our code is available as a Supplementary Material, in order to preserve anonymity.

ACKNOWLEDGMENTS

This work was supported in part by the Alfried Krupp Prize for Young University Teachers of the Alfried Krupp von Bohlen und Halbach-Stiftung (K.B.).

REFERENCES

Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, 13 (5–6):531–712, 2020.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Wacha Bounliphone, Eugene Belilovsky, Matthew B. Blaschko, Ioannis Antonoglou, and Arthur Gretton. A test of relative similarity for model selection in generative models. In *International Conference on Learning Representations*, 2016.

Martin R. Bridson and André Haefliger. The model spaces M_κ^n. In *Metric Spaces of Non-Positive Curvature*, pp. 15–31. Springer, Berlin, Heidelberg, 1999. ISBN 978-3-662-12494-9. doi: 10. 1007/978-3-662-12494-9_2.

Xiaohui Chen, Xu Han, Jiajing Hu, Francisco Ruiz, and Liping Liu. Order matters: Probabilistic modeling of node sequence for graph generation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1630–1639. PMLR, 18–24 Jul 2021.

Fan R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2302–2312. PMLR, 13–18 Jul 2020.

Aasa Feragen, François Lauze, and Søren Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3032–3042, 2015.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 12531263. Association for Computing Machinery, 2020.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J. Smola. A kernel method for the two-sample-problem. In B. Schölkopf, J. C. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems 19*, pp. 513–520. MIT Press, 2007.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012a.

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012b.

Mikhail Gromov. Hyperbolic groups. In S. M. Gersten (ed.), *Essays in Group Theory*, pp. 75–263. Springer, Heidelberg, Germany, 1987.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6626–6637. Curran Associates, Inc., 2017.

Ana Justel, Daniel Peña, and Rubén Zamar. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997. ISSN 0167-7152.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

James R Lloyd and Zoubin Ghahramani. Statistical model criticism using kernel two sample tests. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Lu Mi, Hang Zhao, Charlie Nash, Xiaohan Jin, Jiyang Gao, Chen Sun, Cordelia Schmid, Nir Shavit, Yuning Chai, and Dragomir Anguelov. Hdmapgen: A hierarchical graph generative model of high definition maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4227–4236, June 2021.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4474–4484. PMLR, 26–28 Aug 2020. URL `http://proceedings.mlr.press/v108/niu20a.html`.

Marco Podda and Davide Bacciu. Graphgen-redux: a fast and lightweight recurrent model for labeled graph generation, 2021.

Allen J. Schwenk. Almost all trees are cospectral. In Frank Harary (ed.), *New Directions in the Theory of Graphs*, pp. 275–307. Academic Press, 1973.

Danica J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, , and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *International Conference on Learning Representations*, 2017.

Edwing R. van Dam and Willem H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra and its Applications*, 373:241–272, 2003.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717. PMLR, 2018.

Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, August 2009. doi: 10.14778/1687627.1687631.

Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and Chen Ling. Tg-gan: Continuous-time temporal graph deep generative models with time-validity constraints. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 21042116. Association for Computing Machinery, 2021.

A APPENDIX

The following sections provide additional details about the issues with existing methods. We also show additional plots of our ranking and perturbation experiments.

A.1 KERNELS BASED ON TOTAL VARIATION DISTANCE

Previous work used kernels based on the total variation distance in order to compare evaluation functions via MMD. The choice of this distance, however, requires subtle changes in the selection of kernels for MMD—it turns out that the usual RBF kernel must *not* be used here!

We briefly recapitulate the definition of the total variation distance before explaining its use in the kernel context: given two finite-dimensional real-valued histograms $X := \{x_1, \ldots, x_n\}$ and $Y := \{y_1, \ldots, y_n\}$, their *total variation distance* is defined as

$$\mathrm{d}_{\mathrm{TV}}(X, Y) := \frac{1}{2} \sum_{i=1}^{n} |x_i - y_i| \,. \tag{3}$$

This distance induces a metric space that is not *flat*. In other words, the metric space induced by Eq. 3 has non-zero curvature (a fact that precludes certain kernels from being used together with Eq. 3). We can formalise this by showing that the induced metric space *cannot* have a bound on its curvature.

Theorem 1. *The metric space X_{TV} induced by the total variation distance between two histograms is not in $\mathrm{CAT}(k)$ for $k > 0$, where $\mathrm{CAT}(k)$ refers to the category of metric spaces with curvature bounded from above by k (Gromov, 1987).*

Proof. Let $x_1 = (1, 0, \ldots, 0)$ and $x_2 = (0, 1, 0, \ldots, 0)$. There are at least two geodesics—shortest paths—of the same length, one that first decreases the first coordinate and subsequently increases the second one, whereas for the second geodesic this order is switched. More precisely, the first geodesic proceeds from x_1 to $x_1 - (\epsilon, 0, \ldots, 0)$ for an infinitesimal $\epsilon > 0$, until $x_0 = (0, \ldots, 0)$ has been reached. Following this, the geodesic continues from x_0 to $x_0 + (0, \epsilon, 0, \ldots, 0)$ in infinitesimal steps until x_2 has been reached. The order of these two operations can be switched, such that the geodesic goes from x_1 to $(1, 1, 0, \ldots, 0)$, from which it finally continues to x_2. Both of these geodesics have a length of 1. Since geodesics in a $\mathrm{CAT}(k)$ space for $k > 0$ are *unique* (Bridson & Haefliger, 1999, Proposition 2.11, p. 23), X_{TV} is not in $\mathrm{CAT}(k)$ for $k > 0$. □

Since every $\mathrm{CAT}(k)$ space is also a $\mathrm{CAT}(l)$ space for all $l > k$, this theorem has the consequence that X_{TV} cannot be a $\mathrm{CAT}(0)$ space. Moreover, as every *flat* metric space is in particular a $\mathrm{CAT}(0)$ space, X_{TV} is not flat. According to Theorem 1 of Feragen et al. (2015), the associated geodesic Gaussian kernel, i.e. the kernel that we obtain by writing

$$\mathrm{k}(x, y) := \exp\left(-\frac{\mathrm{d}_{\mathrm{TV}}(X, Y)^2}{2\sigma^2}\right), \tag{4}$$

is *not* positive definite and should therefore not be used with MMD. One potential fix for this specific distance involves using the Laplacian kernel, i.e.

$$\mathrm{k}(x, y) := \exp(-\lambda \, \mathrm{d}_{\mathrm{TV}}(X, Y)). \tag{5}$$

The subtle difference between these kernel functions—only an exponent is being changed—demonstrate that care must be taken when selecting kernels for use with MMD.

A.2 EXPERIMENTAL SETUP

We can analyse the desiderata outlined above using an experimental setting. In the following, we will assess expressivity, robustness, and efficiency for a set of common perturbations, i.e. (i) random edge insertions, (ii) random edge deletions, (iii) random rewiring operations, i.e. 'swapping' edges, and (iv) random node additions. For each perturbation type, we will investigate how the metric changes for an ever-increasing degree of perturbation, where each perturbation is parametrized by at least one parameter. When removing edges, for instance, this is the probability of removing an edge in the graph. Thus for a graph with 100 edges and $p_{\text{remove}} = 0.1$ we would expect on average 90 edges to

remain in the graph. Similar parametrizations apply for the other perturbations, for an more detailed description we refer to Appendix A.3. All these operations are inherently small-scale (though not necessarily localised to specific regions within a graph), but turn into large-scale perturbations of a graph depending on the strength of the perturbation performed. We performed these perturbations using our own `Python`-based framework for graph generative model comparison and evaluation. Our framework additionally permits the simple integration of additional descriptor functions and evaluators.

A.3 Details on graph perturbations

We describe all graph perturbations used in this work on the example of a single graph $G := (V, E)$ where V refers to the vertices of the graph and E to the edges.

Add Edges For each $v_i, v_j \in V$ with $v_i \neq v_j$ a sample from a Bernoulli distribution $x_{ij} \sim \text{Ber}(p_{\text{add}})$ is drawn. Samples for which $x_{ij} = 1$, are added to the list of edges such that $E' = E \cup \{(v_i, v_j) \mid x_{ij} = 1\}$.

Remove Edges For each $e_i \in E$ a sample from a Bernoulli distribution $x_i \sim \text{Ber}(p_{\text{remove}})$ is draw, samples with $x_i = 1$ are removed from the edge list, such that $E' = E \cap \{e_i \mid x_i \neq 1\}$.

Rewire Edges For each $e_i \in E$ a sample from a Bernoulli distribution $x_i \sim \text{Ber}(p_{\text{rewire}})$ is drawn, samples with $x_i = 1$ are rewired. For rewiring a further random variable $y_i \sim \text{Ber}(0.5)$ is drawn which determines which node $e_i[y_i]$ of the edge e_i is kept. The node to which the edge is connected is chosen uniformly from the set of vertices $v_i \in V$, where $v_i \notin e_i$, i.e. avoiding self-loops and reconnecting the original edge. Finally the original edge is removed and the new edge $e'_i = (e_i[y_i], v_i)$ is added to the graph $E' = E \cap \{e_i \mid x_i \neq 1\} \cup \{e'_i \mid x_i = 1\}$.

Add Connected Node We define a set of vertices to be added $V^* = \{v_i \mid |V| < i \leq |V| + n\}$, where n represents the number of nodes to add. For each $v_i \in V$ and $v_j \in V^*$ we draw a sample from a Bernoulli distribution $x_{ij} \sim \text{Ber}(p_{\text{connect_node}})$ and an edge between v_i and v_j to the graph if $x_{ij} = 1$. Thus $E' = E \cup \{(v_i, v_j) \mid v_i \in V, v_j \in V^*, x_{ij} = 1\}$.

A.4 Implementation details

We used the official implementations of GraphRNN, GRAN and Graph Score Matching in our experiments. GraphRNN and GRAN both have an MIT License, and Graph Score Matching is licensed under GNU General Public License v3.0. To preserve anonymity in the review process, we will censor the GitHub link (`https:/www.github.com/BorgwardtLab/ggme`), which will be available upon publication under a BSD 3-Clause license.

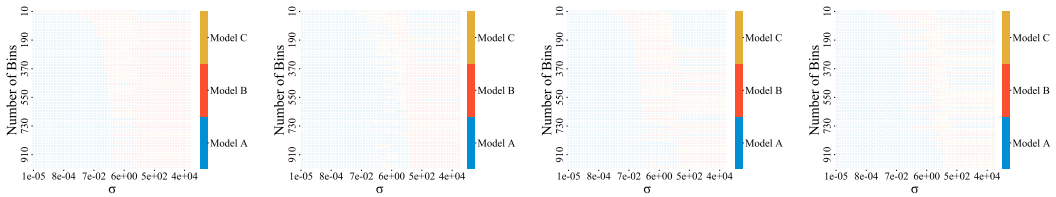
Compute resources. All the jobs were run on our internal cluster, comprising 64 physical cores (`Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz`) with 8 GeForce GTX 1080 GPUs. We stress that the main component of this paper, i.e. the evaluation itself, do *not* necessarily require a cluster environment. The cluster was chosen because individual generative models had to be trained in order to obtain generated graphs, which we can subsequently analyse and rank.

A.5 Speed up trick

As the combination of kernels and hyperparameters yielded hundreds of combinations, it is worth mentioning a worthwhile speedup trick to reduce the complexity of assessing the Gaussian and Laplacian kernel combinations. Considering they have a shared intermediate value, namely the Euclidean distance, it is possible to return intermediate values in the MMD computation (K_{XX}, K_{YY}, and K_{XY}, prior to exponentiation, potential squaring, and scaling by σ. Storing these intermediate results allows one to rapidly iterate over a grid of values for σ without needed to recalculate MMD, leading to a worthwhile speedup.

A.6 EXPERIMENTAL RESULTS

We now provide the results presented in the paper across all the datasets.



(a) Barabási-Albert Graphs (b) Community Graphs (c) Erdös-Rényi Graphs (d) Watts-Strogatz Graphs

Figure 5: A heatmap of which model (from A, B, and C) ranks first in terms of MMD across different hyperparameter combinations. This uses the clustering coefficient descriptor function and the RBF kernel, where the number of bins is a hyperparameter of the descriptor function, and σ is the hyperparameter in the kernel.

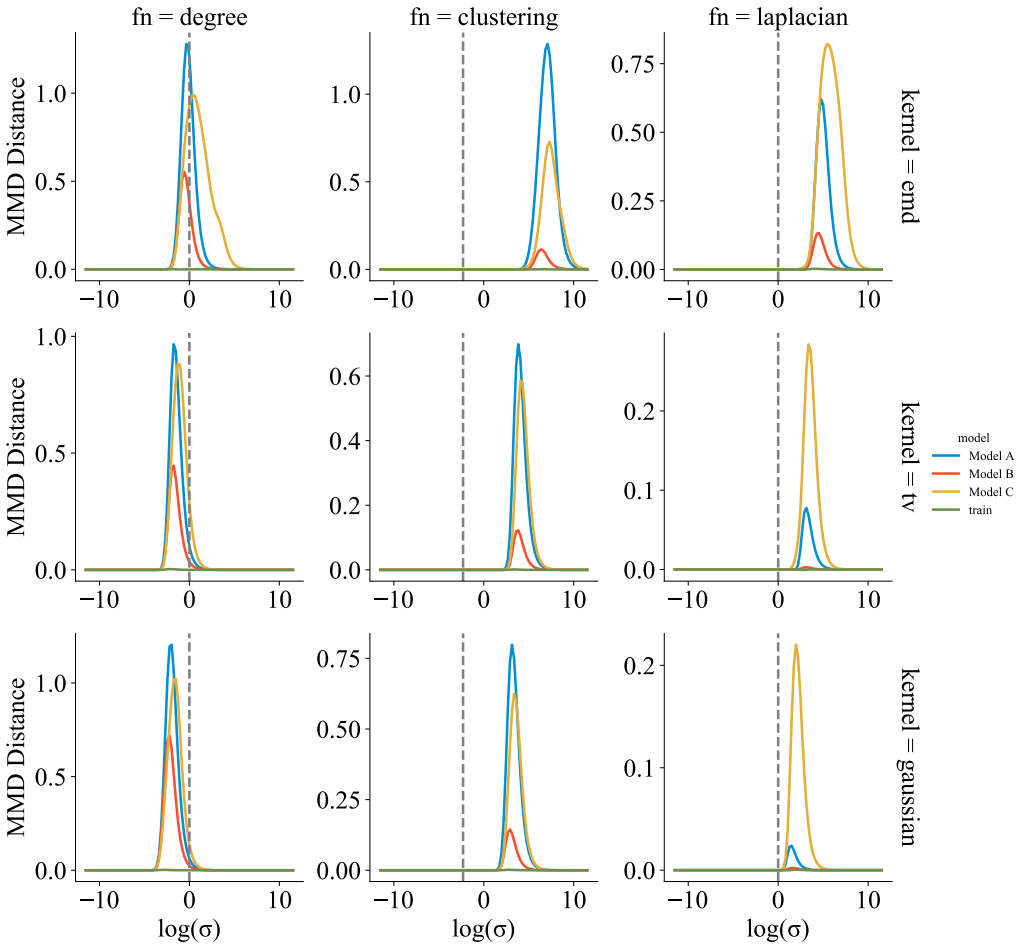


Figure 6: **Barabási-Albert Graphs**. MMD calculated between the test graphs and predictions from three recent graph generative models (A, B, C) over a range of values of σ on the Barabási-Albert Graphs dataset. Additionally, the MMD distance between the test graphs and training graphs is provided to give a meaningful sense of scale to the metric. It provides an idea of what value of MMD signifies an indistinguishable difference between the two distributions.

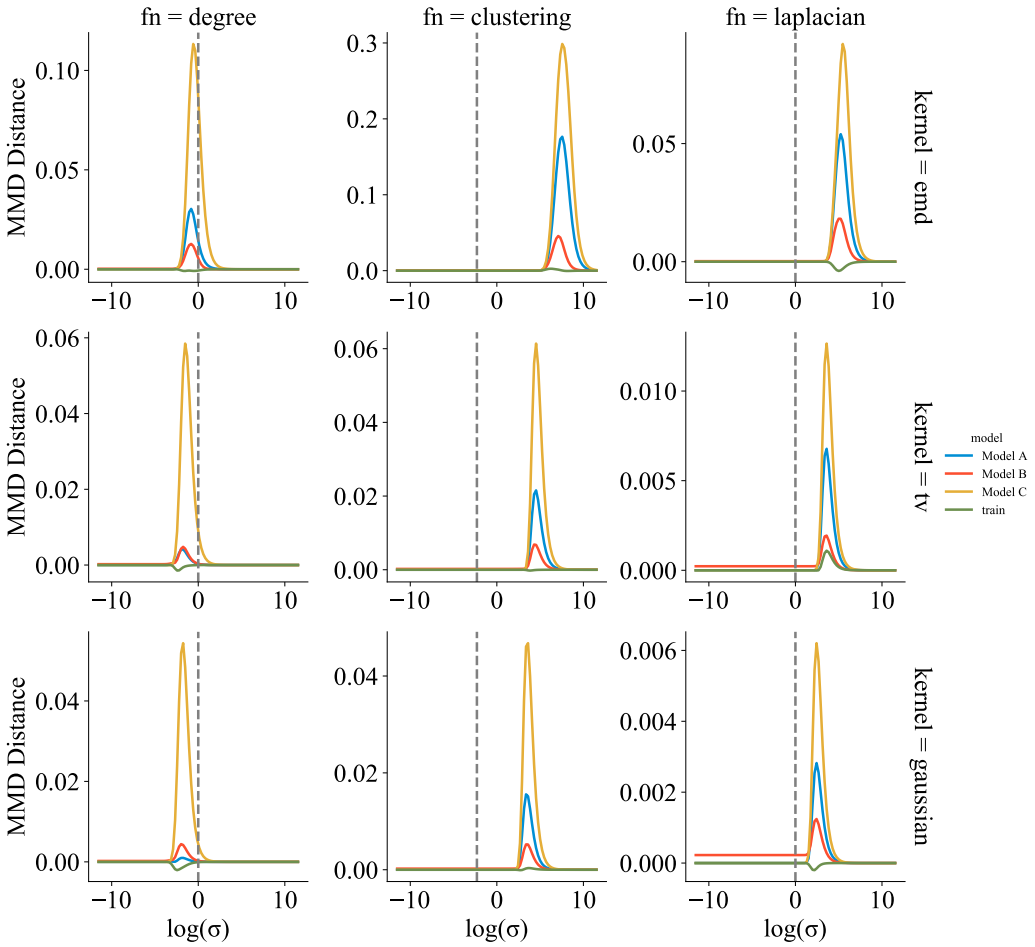


Figure 7: **Community Graphs**. MMD calculated between the test graphs and predictions from three recent graph generative models (A, B, C) over a range of values of σ on the Community Graphs dataset. Additionally, the MMD distance between the test graphs and training graphs is provided to give a meaningful sense of scale to the metric. It provides an idea of what value of MMD signifies an indistinguishable difference between the two distributions.

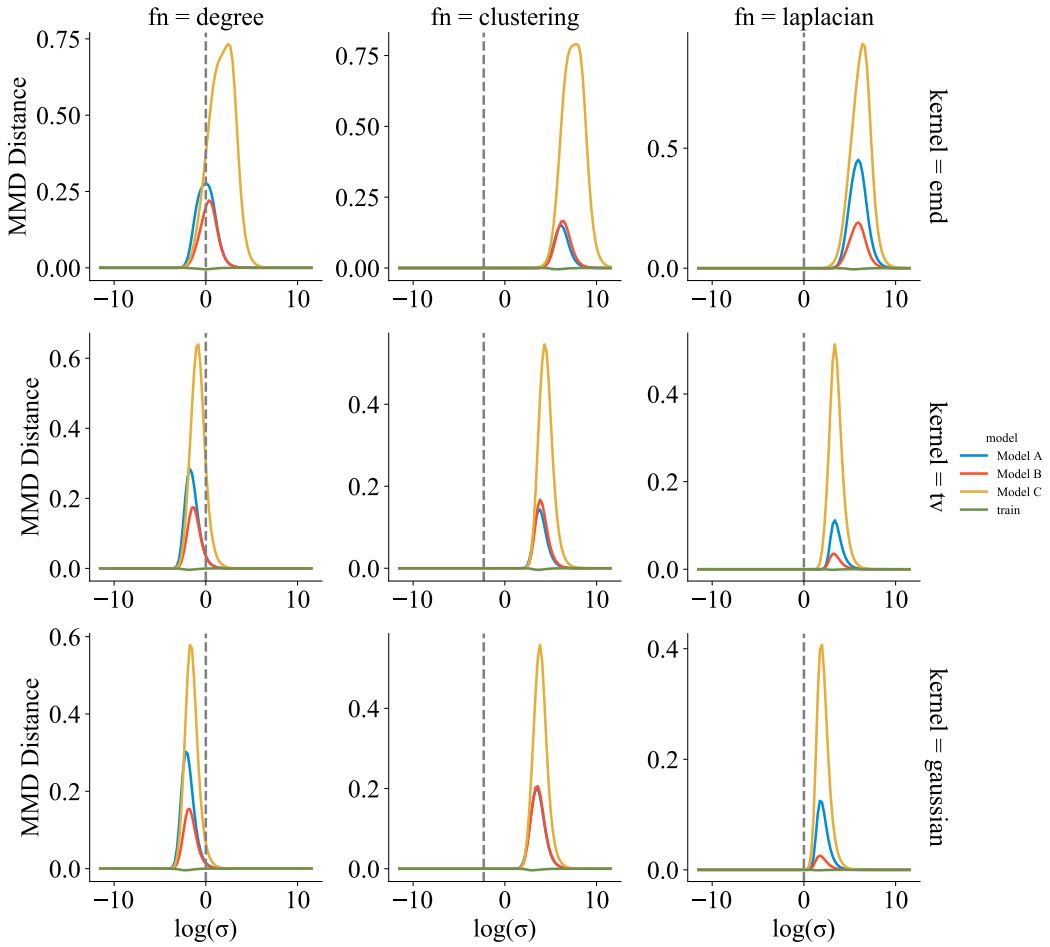


Figure 8: **Erdös-Rényi Graphs**. MMD calculated between the test graphs and predictions from three recent graph generative models (A, B, C) over a range of values of σ on the Erdös-Rényi Graphs dataset. Additionally, the MMD distance between the test graphs and training graphs is provided to give a meaningful sense of scale to the metric. It provides an idea of what value of MMD signifies an indistinguishable difference between the two distributions.

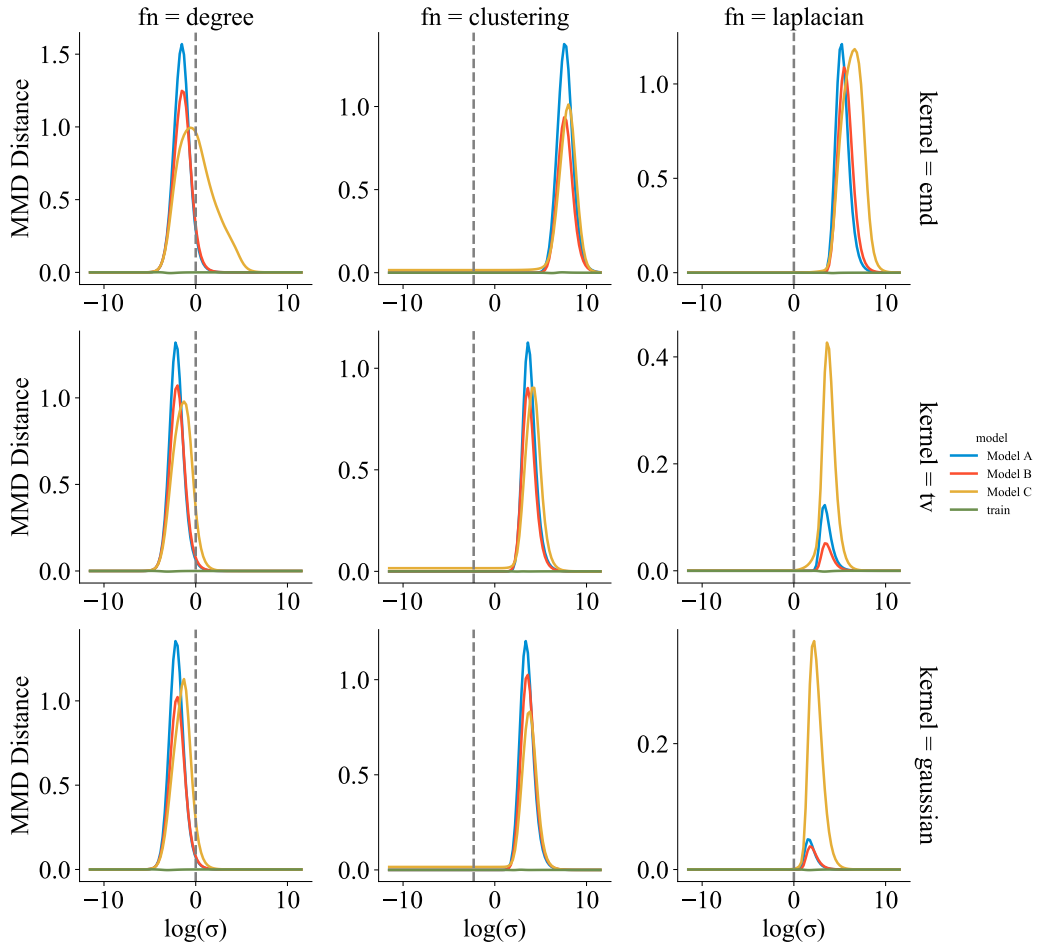


Figure 9: **Watts-Strogatz Graphs**. MMD calculated between the test graphs and predictions from three recent graph generative models (A, B, C) over a range of values of σ on the Watts-Strogatz Graphs dataset. Additionally, the MMD distance between the test graphs and training graphs is provided to give a meaningful sense of scale to the metric. It provides an idea of what value of MMD signifies an indistinguishable difference between the two distributions.

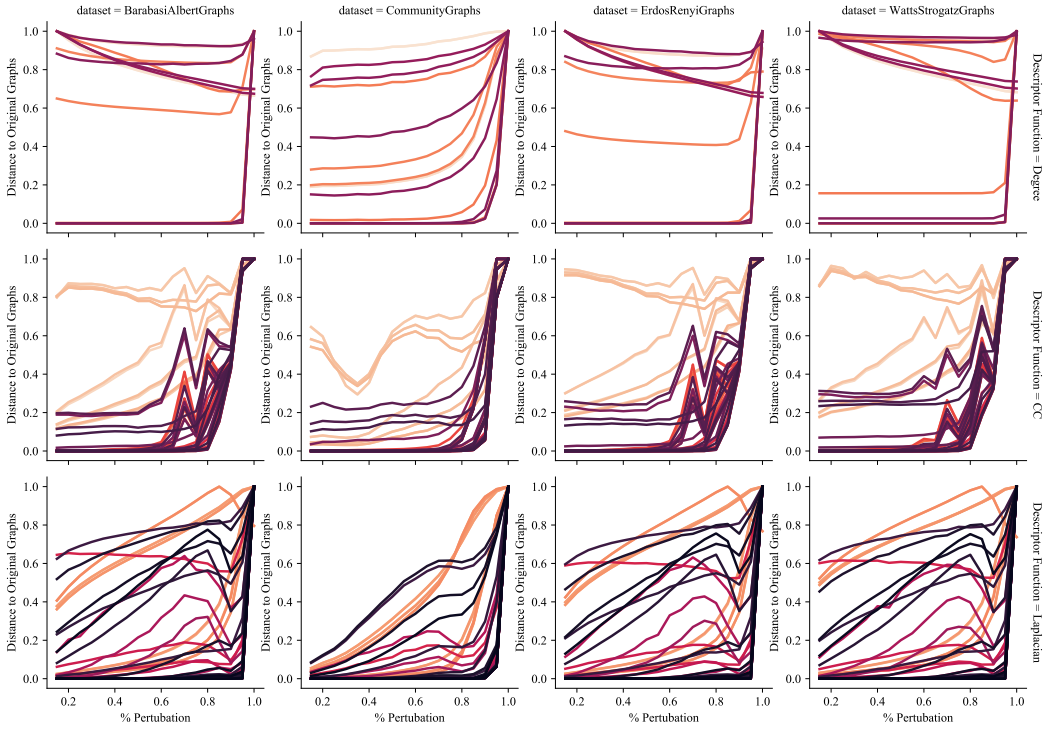


Figure 10: **Perturbation: adding edges**. This figure shows the full results across datasets, descriptor functions and parameters when the perturbation is adding edges to the graphs. At each level of perturbation, the distance of the perturbed graphs is calculated to the original graph distribution using the specified evaluator function. Each line represents a different parameter combination. An ideal evaluator function would monotonically increase as the degree of perturbation increases.

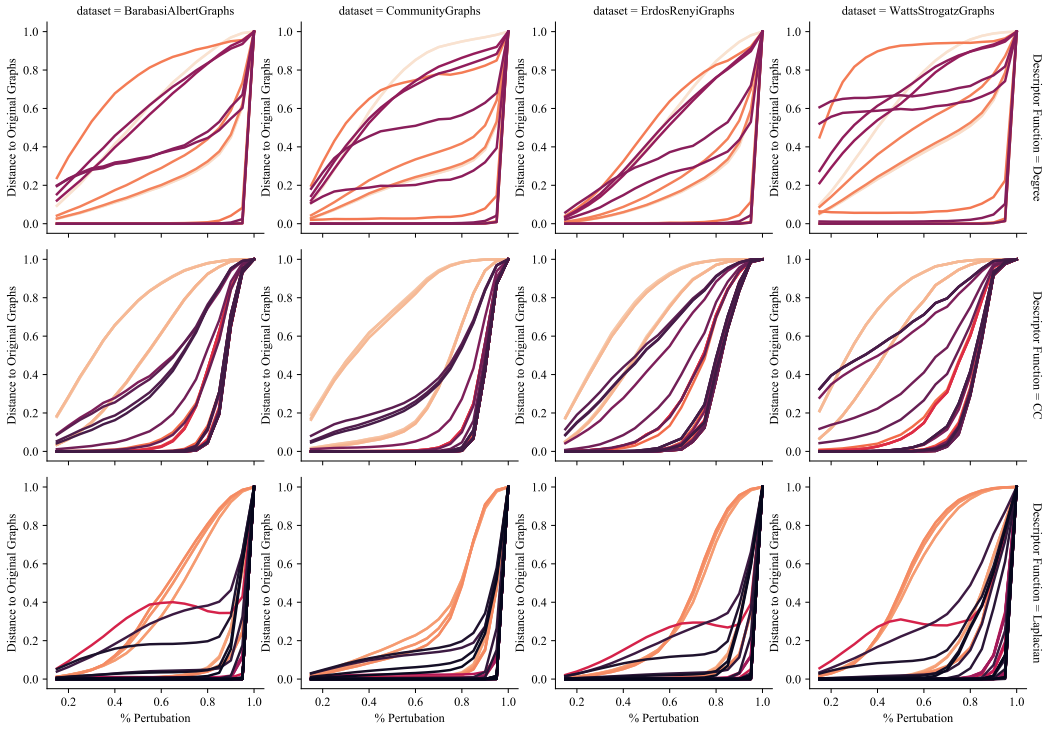


Figure 11: **Perturbation: removing edges**. This figure shows the full results across datasets, descriptor functions and parameters when the perturbation is removing edges from the graphs. At each level of perturbation, the distance of the perturbed graphs is calculated to the original graph distribution using the specified evaluator function. Each line represents a different parameter combination. An ideal evaluator function would monotonically increase as the degree of perturbation increases.

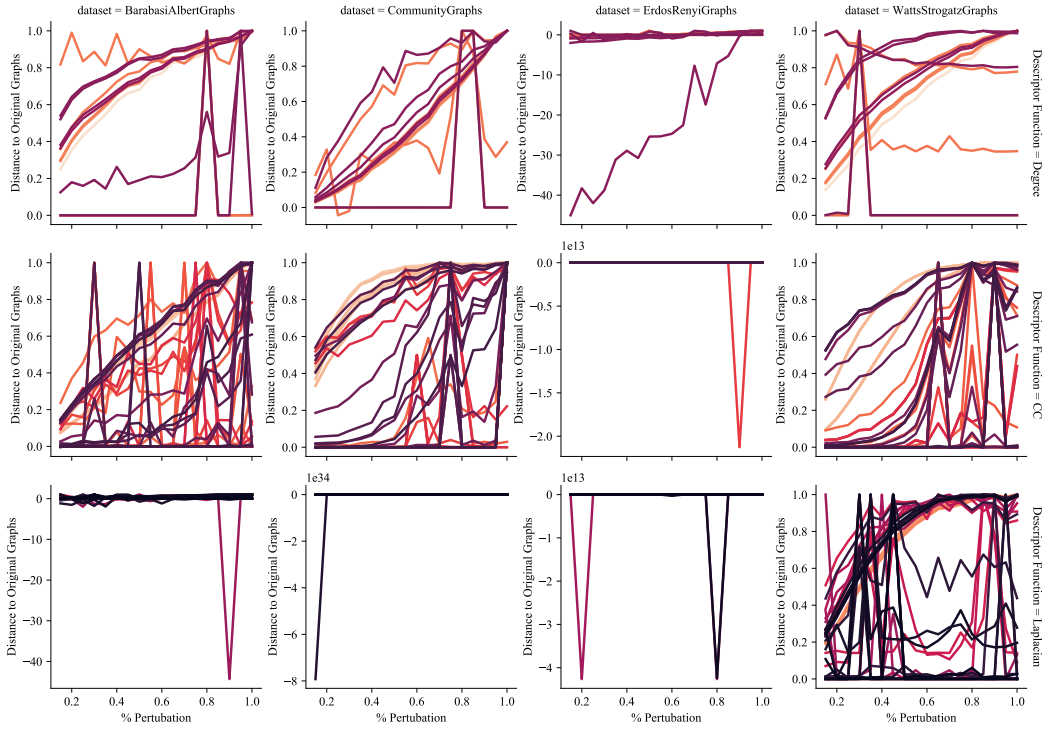


Figure 12: **Perturbation: rewiring edges**. This figure shows the full results across datasets, descriptor functions and parameters when the perturbation is rewiring edges in the graphs. At each level of perturbation, the distance of the perturbed graphs is calculated to the original graph distribution using the specified evaluator function. Each line represents a different parameter combination. An ideal evaluator function would monotonically increase as the degree of perturbation increases.

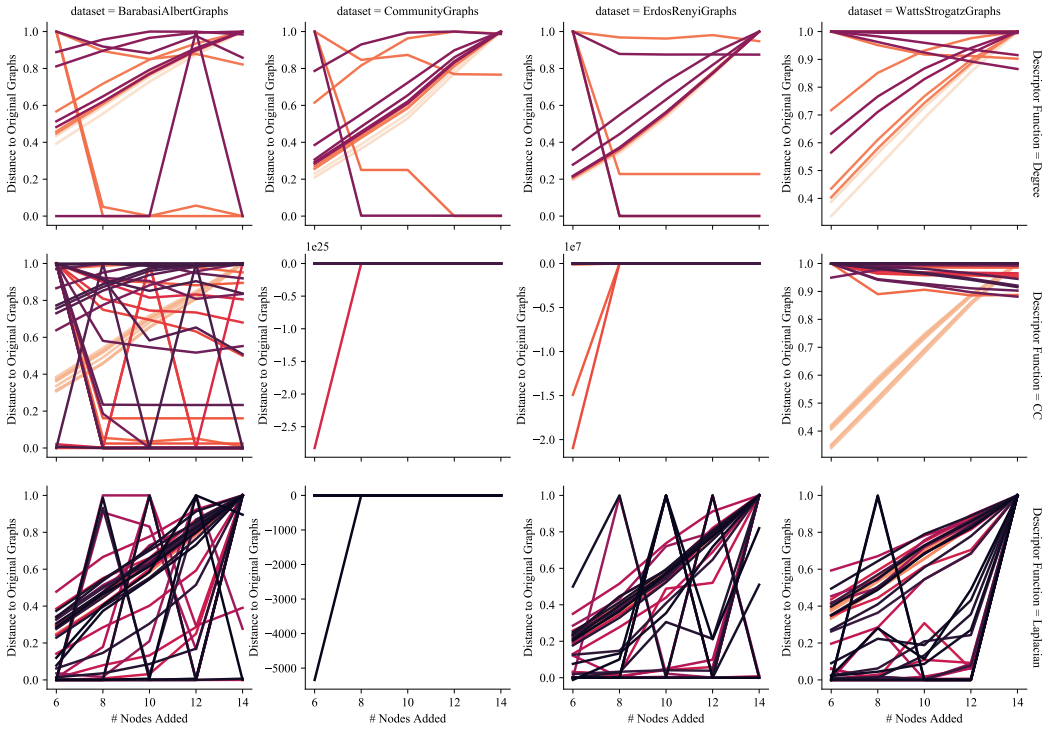
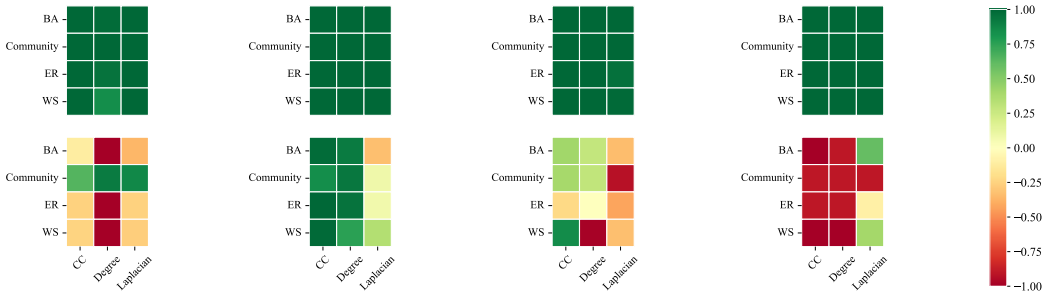


Figure 13: **Perturbation: adding connected nodes**. This figure shows the full results across datasets, descriptor functions and parameters when the perturbation is adding connected nodes to the graphs (for each node that is added, there is a 15% chance the node will be connected to any other node in the graph). At each level of perturbation, the distance of the perturbed graphs is calculated to the original graph distribution using the specified evaluator function. Each line represents a different parameter combination. An ideal evaluator function would monotonically increase as the degree of perturbation increases.

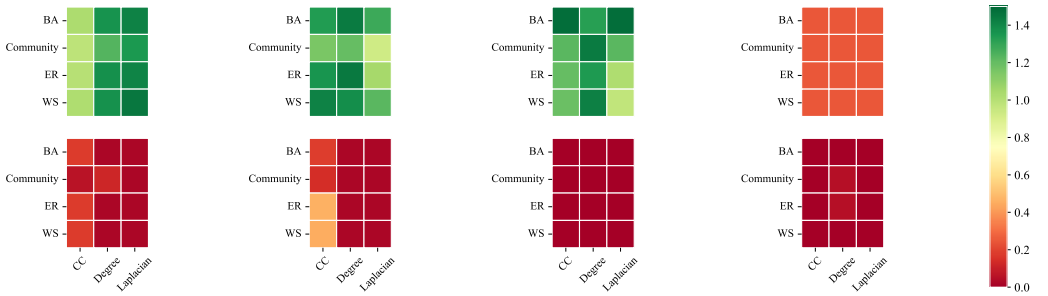
A.7 ALTERNATIVE MEASURES OF CORRELATION

As a general recommendation, we chose to use the Pearson correlation coefficient to select a good kernel-hyperparameter combination, due to its simplicity and ease of interpretation. It reflects the behavior we expect from perturbations: as the graphs are increasingly perturbed, the distance to the original graphs should in a similar manner grow as well. However, there could be scenarios in which the distance to the original graphs should not grow linearly with the degree of perturbation, in which case the Pearson correlation coefficient would not be the best choice. We present here two plug-in alternatives for the Pearson correlation coefficient, namely the Spearman rank correlation coefficient, and mutual information, which can easily be integrated into our framework. We add one word of caution when using the mutual information, which is the fact that it does not capture the directionality of dependence. This is only an issue if there are scenarios in which the MMD distance *decreases* as the degree of perturbation increases. Since we observed this in some of our datasets, it would not be the most appropriate to use in this specific case. We now present the results from using these measures of dependence below.



(a) AddEdges (b) RemoveEdges (c) RewireEdges (d) AddConnectedNodes

Figure 14: The Spearman rank correlation of MMD with the degree of perturbation in the graph, assessed for different descriptor functions and datasets (BA: Barabási-Albert Graphs, ER: Erdös-Rényi Graphs, WS: Watts-Strogatz Graphs). For an ideal metric, the distance would increase as the degree of perturbation increases; resulting in a correlation close to 1. The upper row shows the *best* kernel-parameter combination in terms of the correlation; the bottom row shows the *worst*. As we can see, a proper kernel and parameter selection leads to strong correlation to the perturbation, but a bad choice can lead to inverse correlation, highlighting the importance of a good kernel/parameter combination.



(a) AddEdges (b) RemoveEdges (c) RewireEdges (d) AddConnectedNodes

Figure 15: The mutual information of MMD with the degree of perturbation in the graph, assessed for different descriptor functions and datasets (BA: Barabási-Albert Graphs, ER: Erdös-Rényi Graphs, WS: Watts-Strogatz Graphs). For an ideal metric, the distance would increase as the degree of perturbation increases; resulting in a high mutual information coefficient. The upper row shows the *best* kernel-parameter combination in terms of the mutual information; the bottom row shows the *worst*. As we can see, a proper kernel and parameter selection leads to strong dependence on the perturbation, but a bad choice can lead to no mutual information, highlighting the importance of a good kernel/parameter combination.

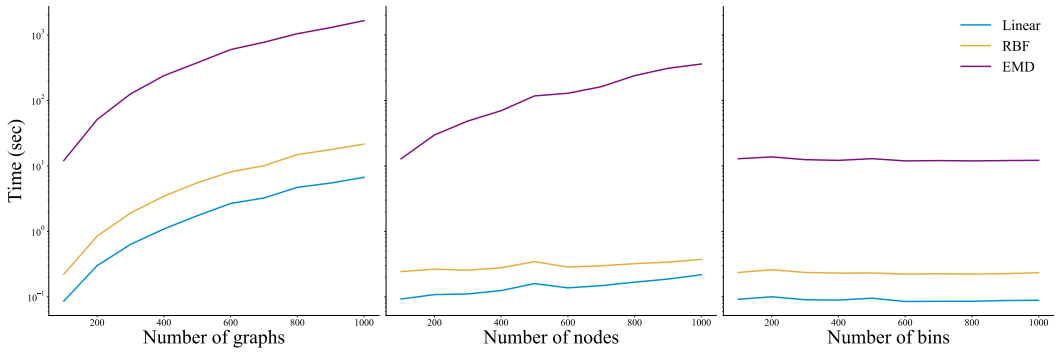


Figure 16: CPU runtime comparison (lower is better) of the linear, RBF, and EMD-based kernels when evaluated on ER graphs and varying the number of graphs (the dataset size), the number of nodes (the size of each graph), and histogram bin size. Each plot varies a single parameter (on the x-axis), while keeping the other two fixed with values of 100. Runtimes are reported on a logarithmic scale.

A.8 Computational runtime of different kernels

In the following, we assess the empirical CPU runtime of the linear kernel, RBF kernel, and the EMD-based kernel in the MMD calculation. As noted by ourselves and Liao et al. (2019), the EMD-based kernel is computationally expensive, hindering its suitability for use in graph generative model evaluation. We investigate this effect in more detail by doing a runtime comparison of the three kernels in a simulated environment. We generate two sets of Erdös-Rényi Graphs with the probability of an edge $p = 0.3$, and then calculate the MMD distance between the two sets using the degree distribution as the descriptor function for a varying dataset size (n_{graphs}), graph size (n_{nodes}), and histogram bin size (n_{bins}). As a default, we set $n_{\text{graphs}} = 100$, $n_{\text{nodes}} = 100$, and $n_{\text{bins}} = 100$. We then change one variable at a time, iterating through values $\{100, 200, \ldots, 1000\}$ while keeping the other two variables fixed, and measured the time it took to calculate the MMD distance. We report the average runtime over ten repetitions to obtain more stable results.

Our results can be seen in Figure 16. At the default setting of 100 graphs, 100 nodes per graph, and 100 bins in the histogram, we observed that the EMD-based kernel took more than 50-140 times longer to compute compared to the RBF and linear kernels respectively. This difference was exacerbated when the size of the dataset (n_{graphs}) increased as well as when the size of the graphs in the dataset (n_{nodes}) increased. For a dataset of 1,000 graphs, the EMD-based kernel took 27 minutes to run, whereas the linear kernel took 6 seconds and the RBF kernel took 21 seconds. Doing a simple extrapolation of the runtime of a dataset comprising 10,000 graphs, we estimate the EMD-based kernel would take 50 hours to run for a single hyperparameter combination, showcasing the computational limitation of this choice of kernel for larger dataset sizes. We observed a similar, albeit less extreme, increase in runtime when the size of the graphs increased. While the linear time approximation of MMD (Gretton et al., 2012a) could mitigate some of the runtime challenges, in general we recommend that practitioners use efficient kernels such as the linear kernel or RBF kernel for graph generative model evaluation.