AgraSSt: Approximate Graph Stein Statistics for Interpretable Assessment of Implicit Graph Generators

Wenkai Xu
Department of Statistics
University of Oxford
wenkai.xu@stats.ox.ac.uk

Gesine Reinert
Department of Statistics
University of Oxford
and Alan Turing Institute
reinert@stats.ox.ac.uk

Abstract

We propose and analyse a novel statistical procedure, coined *AgraSSt*, to assess the quality of graph generators which may not be available in explicit forms. In particular, AgraSSt can be used to determine whether a *learned* graph generating process is capable of generating graphs which resemble a given input graph. Inspired by Stein operators for random graphs, the key idea of AgraSSt is the construction of a kernel discrepancy based on an operator obtained from the graph generator. AgraSSt can provide interpretable criticisms for a graph generator training procedure and help identify reliable sample batches for downstream tasks. We give theoretical guarantees for a broad class of random graph models. Moreover, we provide empirical results on both synthetic input graphs with *known* graph generation procedures, and real-world input graphs that the state-of-the-art (deep) generative models for graphs are *trained* on.

1 Introduction

Generative models for graphs have received increasing attention in the statistics and machine learning communities. Recently, deep neural networks have been utilised to learn rich representations from graph structures and generate graphs [Dai et al., 2020, Li et al., 2018, Liao et al., 2019, You et al., 2018]. However, due to the often opaque deep learning procedures, these deep generative models are usually implicit, which hinders theoretical analysis to assess how close the generated samples are in their distributional properties to the graph distribution they are meant to be sampled from.

Learning instead parametric models from an explicit pre-specified probability distribution class would allow to use the learned parameters for model assessment. However, parametric models may only capture a fraction of the graph features and have restricted modelling power. Parameter estimation can be inconsistent [Shalizi and Rinaldo, 2013] even for well-specified models, and may lead to wrong conclusions in model assessments. Using instead deep generative models for graphs may surpass some of these issues by learning rich graph representations, but methods to assess the quality of such implicit graph generators are lacking. In principle, nonparametric hypothesis tests can be useful to assess complex models, such as kernel-based tests procedures that utilise functions in a reproducing kernel Hilbert space (RKHS) [Berlinet and Thomas, 2004]. When the models are described in the form of explicit probabilities, goodness-of-fit tests [Chwialkowski et al., 2016, Liu et al., 2016] may apply; however, goodness-of-fit testing procedures are generally not applicable for implicit models. Instead, if a large set of samples from the target distribution is observed, one may generate samples from the implicit model and perform a two-sample test such as a maximum mean discrepancy (MMD) test [Gretton et al., 2007] for model assessment [Jitkrittum et al., 2017, Xu and Matsuda, 2020, 2021].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

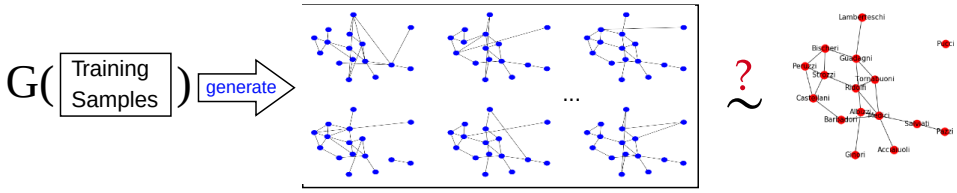


Figure 1: Illustration for the assessment task: a graph generator G, which is learned from training samples, generates a set of network samples (vertices in blue), and is assessed against the target graph, which here is Padgett's Florentine marriage network (with vertices in red labelled as family names).

Yet, in real-world applications, often only a single graph from the target distribution is observed [Bresler and Nagaraj, 2018, Reinert and Ross, 2019]. In this case, the MMD methods for model assessment via two-sample testing procedures cannot be used to assess implicit graph generators. To our knowledge, beyond simple Monte Carlo tests with often poor power, no principled test is available for assessing the quality of implicit graph generators. Figure.1 illustrates the task: a given graph generator G which is learned from a set of training samples can generate samples of user-defined size. The task is to assess whether G can generate samples from the same distribution that generates the observed graph, for example, Padgett's Florentine marriage network [Padgett and Ansell, 1993]. The samples in Figure.1 are generated by a Cross-Entropy Low-rank Logit (CELL) model [Rendsburg et al., 2020] trained on Padgett's Florentine marriage network, but any graph generator could be used.

This paper makes three main contributions. (1) We introduce Approximate graph Stein Statistics (AgraSSt) in Section 3, which opens up a *principled way* to understand implicit graph generators. AgraSSt is a variant of a kernel Stein discrepancy based on an empirical Stein operator for conditional distributions of general random graph models. The testing procedure is inspired by gKSS, a goodness-of-fit testing procedure for explicit exponential random graph models (ERGMs) [Xu and Reinert, 2021]. (2) We provide theoretical guarantees for AgraSSt (Section 3). (3) We propose interpretable model criticisms when there is a model misfit, and we identify representative synthetic sample batches when the model is a good fit (Section 4). Further, in Section 2 we review gKSS. We provide empirical results in Section 5 and a discussion in Section 6. Proof details, more background, theoretical and empirical results, and implementation details are found in the Supplementary Information (SI).The code for the experiments is available at `https://github.com/wenkaixl/agrasst`.

2 Background: graph kernel Stein statistics

In [Xu and Reinert, 2021], a goodness-of-fit testing procedure called gKSS for exponential random graph models is introduced, which assumes that only a single network may be observed in the sample. As AgraSSt is inspired by gKSS, and gKSS serves as comparison method when the observed network is known to be generated from an exponential random graph model (ERGM), we briefly review gKSS. The notation introduced in this section is used throughout the paper.

ERGMs are extensively used for social network analysis [Wasserman and Faust, 1994, Holland and Leinhardt, 1981, Frank and Strauss, 1986]; a special case are Bernoulli random graphs. ERGMs model random graphs via a Gibbs measure with respect to a chosen set of network statistics, such as number of edges, 2-stars, and triangles. Denote by $\mathcal{G}_n^{lab}$, the set of vertex-labeled graphs on n vertices, with $N = n(n-1)/2$ possible undirected edges. Encode $x \in \mathcal{G}_n^{lab}$ by an ordered collection of $\{0,1\}$-valued variables $x = (x^{(ij)})_{1 \le i < j \le n} \in \{0,1\}^N$ where $x^{(ij)} = 1$ if and only if there is an edge between i and j. We denote an (ordered) vertex-pair index $s = (i,j)$ by $s \in [N] := \{1, \dots, N\}$. Let $t_1(x)$ count the number of edges in x and let $t_2(x), \dots, t_k(x)$ count other small graphs on at most n vertices in x (possibly scaled; for details including a precise definition, Definition A.3, see SI A.2). For $\beta = (\beta_1, \dots, \beta_k)^\top \in \mathbb{R}^k$ and $t(x) = (t_1(x), \dots, t_k(x))^\top \in \mathbb{R}^k$ we say that $X \in \mathcal{G}_n^{lab}$ follows the exponential random graph model $X \sim \mathrm{ERGM}(\beta, t)$ if for $\forall x \in \mathcal{G}_n^{lab}$,

$$q(X = x) = \frac{1}{\kappa_n(\beta)} \exp\left(\sum_{l=1}^{k} \beta_l t_l(x)\right). \tag{1}$$

Here $\kappa_n(\beta)$ is a normalisation constant. In this model, $t_\ell, \ell = 1, \ldots, k$, are sufficient statistics. Parameter estimation $\hat{\beta}_l$ for β_l is *only* possible when the sufficient statistics are specified a priori; see SI.B.1 for estimation details. In modern graph learning procedures, e.g. deep generative learning, the sufficient statistics t_l of Eq.(1) may not be obtained explicitly.

In [Reinert and Ross, 2019] the exponential random graph distribution in Eq. (1) is characterised by a so-called *Stein operator*, as follows. Let $e_s \in \{0,1\}^N$ be a vector with 1 in coordinate s and 0 in all others; $x^{(s,1)} = x + (1 - x_s)e_s$ has the s-entry replaced of x by the value 1, and $x^{(s,0)} = x - x_s e_s$ has the s-entry of x replaced by the value 0; moreover, x_{-s} is the set of edge indicators with entry s removed. For a function $h : \{0,1\}^N \to \mathbb{R}$, let $\Delta_s h(x) = h(x^{(s,1)}) - h(x^{(s,0)})$. Set $q_X(x^{(s,1)}|x_{-s}) = \mathbb{P}(X^s = 1|X_{-s} = x_{-s})$. Define the operator

$$\mathcal{A}_{\beta,t} f(x) = \frac{1}{N} \sum_{s \in [N]} \mathcal{A}_q^{(s)} f(x), \quad \mathcal{A}_q^{(s)} f(x) = q(x^{(s,1)}|x_{-s}) \Delta_s f(x) + \Big(f(x^{(s,0)}) - f(x) \Big). \quad (2)$$

Then under mild conditions Reinert and Ross [2019] show that if $\mathbb{E}_p[\mathcal{A}_{\beta,t} f] = 0$ for all smooth test functions f, then p must be the distribution of ERGM(β, t). Thus, this operator characterises ERGM(β, t). For the derivation of AgraSSt it is of interest to see how this operator is obtained. It is indeed the generator of a so-called *Glauber* Markov chain on $\mathcal{G}_n^{lab}$ with transition probabilities

$$\mathbb{P}(x \to x^{(s,1)}) = N^{-1} - \mathbb{P}(x \to x^{(s,0)}) = N^{-1} q_X(x^{(s,1)}|x).$$

With the ERGM Stein operator in Eq.(2) and a rich-enough RKHS test function class $\mathcal{H}$, Xu and Reinert [2021] propose a graph kernel Stein statistics (gKSS) to perform goodness-of-fit testing on an *explicit* ERGM when a single network sample is observed. With the summand components in Eq.(2), the Stein operator can be seen as taking expectation over vertex-pair variables $S \in [N]$ with uniform probability $\mathbb{P}(S = s) \equiv N^{-1}$ independently of x, namely

$$\mathcal{A}_q f(x) = \sum_{s \in [N]} \mathbb{P}(S = s) \mathcal{A}_q^{(s)} f(x) =: \mathbb{E}_S[\mathcal{A}_q^{(S)} f(x)].$$

For a fixed graph x, gKSS is defined as

$$\text{gKSS}(q; x) = \sup_{\|f\|_{\mathcal{H}} \le 1} \Big| \mathbb{E}_S[\mathcal{A}_q^{(S)} f(x)] \Big|, \quad (3)$$

where the function f is chosen to best distinguish q from x. For an RKHS $\mathcal{H}$ associated with kernel K, by the reproducing property of $\mathcal{H}$, the squared version of gKSS admits a quadratic form representation $\text{gKSS}^2(q; x) = \langle \mathbb{E}_S[\mathcal{A}_q^{(S)} K(x, \cdot)], \mathbb{E}_S[\mathcal{A}_q^{(S)} K(x, \cdot)] \rangle$, which can be computed readily. More background can be found in SI B.2.

3 AgraSSt: Approximate Graph Stein Statistic

An implicit graph generator may not admit a probability distribution in the form of Eq.(1). However, the idea of constructing Stein operators based on Glauber dynamics using conditional probability distribution for ERGM serves as a motivation for us to propose Stein operators for conditional graph distributions, to facilitate an (approximate) characterisation for implicit random graph models.

3.1 Stein operators for conditional graph distributions

Let $q(x) = \mathbb{P}(X = x)$ be any distribution with support $\mathcal{G}_n^{lab}$. Let $t(x)$ denote a statistic on graphs which takes on finitely many values $\underline{k}$ and let $q_{\underline{k}}(x) = \mathbb{P}(X = x|t(x) = \underline{k})$. We assume that $q_{\underline{k}}(x) > 0$ for all $\underline{k}$ under consideration. For a generic outcome we write q_t. We introduce a Markov chain on $\mathcal{G}_n^{lab}$ which transitions from x to $x^{(s,1)}$ with probability

$$q(x^{(s,1)}|t(x_{-s})) = \mathbb{P}(X^s = 1|t(x_{-s})) =: q_t(x^{(s,1)}), \quad (4)$$

and which transitions from x to $x^{(s,0)}$ with probability $q(x^{(s,0)}|t(x_{-s})) = 1 - q_t(x^{(s,1)})$; no other transitions occur. Let

$$\mathcal{A}_{q,t}^{(s)} f(x) = q_t(x^{(s,1)}) f(x^{(s,1)}) + q_t(x^{(s,0)}) f(x^{(s,0)}) - f(x). \quad (5)$$

Algorithm 1 Estimating the conditional probability

Input: Graph generator G; statistics $t(x)$;
Procedure:
1: Generate samples $\{x_1, \ldots, x_L\}$ from G.
2: For $s \in [N], i \in [n]$, let $n_{s,k}$ be the number of graphs $x_l, l \in [L]$, in which s is present and $t(x_{-s}) = k$.
3: Estimate the conditional probability of the edge s being present conditional on $t(x_{-s}) = k$ by an estimator $\widehat{g}_t(s; k)$, using a look-up table or smoothing.
Output: $\widehat{g}_t(s; k)$ that estimates $q(x^{(s)} = 1 | t(x_{-s}) = k)$.

For an ERGM, $t(x)$ could be taken as a sufficient vector of statistics[1], but here we do not assume a parametric network model $q(x)$, and $t(x)$ does not have to be sufficient statistics for $q(x)$.

Recall that an operator is a Stein operator for a distribution μ if its expectation under μ is zero. The following result provides a theoretical foundation for AgraSSt and is proven in SI.A.

Lemma 3.1. *If $q_{\underline{k}}(x) > 0$ for all $\underline{k}$, then $\mathcal{A}^{(s)}_{q,t=\underline{k}}$ is a Stein operator for the conditional distribution of X given $t(X) = \underline{k}$, and $\sum_s \mathcal{A}^{(s)}_{q,t=\underline{k}}$ is a Stein operator for the conditional distribution of X given $t(X) = \underline{k}$.*

In particular, $\mathbb{E}_{q_{t=\underline{k}}}[\mathcal{A}^{(s)}_{q,t=\underline{k}}] = 0$. Intuitively, if the distribution of $\tilde{X}$ is close to that of X, then with $\tilde{Y}_{t=\underline{k}}$ denoting the corresponding random graph with distribution that of $\tilde{X}$ given $t(\tilde{X}) = \underline{k}$, it should hold that $\mathbb{E}[\mathcal{A}^{(s)}_{q,t=\underline{k}}(\tilde{Y}_{t=\underline{k}})] \approx 0$. In this way the Stein operator in Eq.(5) can be used to assess similarity between distributions.

3.2 Approximate Stein operators

For implicit models and graph generators G, the Stein operator $\mathcal{A}^{(s)}_{q,t}$ in Eq.(5) cannot be obtained without explicit knowledge of $q_t(x^{(s,1)})$. However, given a large number for samples from the graph generator G, the conditional edge probabilities $q_t(x^{(s,1)})$ can be estimated. Here we denote by $\widehat{q}_t(x^{(s,1)})$ an estimate of $q_t(x^{(s,1)})$; some estimators will be suggested in Section 3.3.

AgraSSt performs model assessment using an operator which approximates the Stein operator $\mathcal{A}^{(s)}_{q,t}$. We define the approximate Stein operator for the conditional random graph by

$$\mathcal{A}^{(s)}_{\widehat{q},t} f(x) = \widehat{q}_t(x^{(s,1)}) f(x^{(s,1)}) + \widehat{q}_t(x^{(s,0)}) f(x^{(s,0)}) - f(x). \tag{6}$$

The vertex-pair averaged approximate Stein operator is

$$\mathcal{A}_{\widehat{q},t} f(x) = \frac{1}{N} \sum_{s \in [N]} \mathcal{A}^{(s)}_{\widehat{q},t} f(x). \tag{7}$$

3.3 Estimation with chosen statistics on graphs

Using the Stein operator for conditional graph distributions, we can obtain the approximate Stein operators in Eq. (6) and Eq. (7) for an implicit graph generator G by estimating $q_t(x^{(s,1)})$. Here $t(x)$ are user-defined statistics. In principle, any multivariate statistic $t(x)$ can be used in this formalism. However, estimating the conditional probabilities using relative frequencies can be computationally prohibitive when the graphs are very large and specific frequencies are rarely observed. Instead, here we consider simple summary statistics, such as edge density, degree statistics or the number of neighbours connected to both vertices of s. The estimation procedure is presented in Algorithm 1.

To estimate $q_t(x^{(s,1)})$ in Step 3 of Algorithm 1, a look-up table can be used: If $t(x)$ is a possibly multivariate statistic with a discrete number of outcomes, generate many independent copies from the

[1] When conditioning on the sufficient statistics, for ERGMs the resulting Stein operator allows to establish elegant approximation results [Bresler and Nagaraj, 2019, Reinert and Ross, 2019].

synthetic graph generator, and count $n(\underline{k}, s)$, the number of times that vertex-pair s is present in the simulated graphs and $t(x_{-s}) = \underline{k}$. Consistency follows from the law of large numbers. When using smoothing to estimate the conditional probability, consistency will depend on the smoothing method. If the underlying graph has exchangeable edge indicators then $q_t(x^{(s,1)})$ does not depend on the choice of vertex-pair s. Then we set set $n(\underline{k}) = \sum_s n(\underline{k}, s)$, and $N_{\underline{k}} = \sum_{i=1}^{L} \sum_s \mathbf{1}(t(x_{-s}) = k)$; if $t = \underline{k}$ we estimate $\widehat{q}_t(x^{(s,1)})$ by

$$\hat{g}_t(\underline{k}) = \frac{n_k}{N_{\underline{k}}} \mathbb{1}(N_{\underline{k}} \geq 1). \tag{8}$$

If k_{min}, k_{max} denote the minimum and maximum values of statistics from simulated graph samples, then for $\underline{k}$ outside this set, the lookup table estimator Eq.(8) is set to estimate $\hat{g}(\underline{k}) = 0$.

If the underlying graph cannot be assumed to have exchangeable edge indicators or if the statistic t is high dimensional, then any particular $n(\underline{k}, s)$ may not be observed very often. In such a situation we can learn $\hat{g}_t(s; k)$ using kernel ridge regression so that the conditional probabilities for similar $t(x_{-s})$ are predicted in a smooth manner. Estimating $q(x^s | t(x_{-s}) \leq k)$ instead of $q(x^s | t(x_{-s}) = k)$ may provide an alternative, smoother estimate for the conditional probabilities.

The next result shows that the approximate Stein operator achieves the Stein identity asymptotically. For this result, which is proved in SI.A, we use the notation $\|\Delta f\| = \sup_{s \in [N], x} \|\Delta_s f(x)\|$.

Theorem 3.2. *Assume $\widehat{q}_t(x^{(s,1)})$ is a consistent estimator for $q_t(x^{(s,1)})$ as $L \to \infty$. Then for any f such that $||\Delta f|| < \infty$ we have $\mathbb{E}_q[\mathcal{A}_{\widehat{q},t} f(x)] \to \mathbb{E}_q[\mathcal{A}_{q,t} f(x)] = 0$ as $L \to \infty$.*

Proofs and additional analysis are included in SI A. Proposition A.1 in SI A.1 addresses consistent estimation of $\widehat{q}_t$ and SI A.2 provides refined results for ERGMs, including a Gaussian approximation.

3.4 AgraSSt for implicit graph generators

The estimated conditional probabilities give an approximate Stein operator for Eq.(5). With the appropriately defined Stein operator from an implicit model given in Eq.(6), we can define AgraSSt, a kernel-based statistic analogous to gKSS in Eq.(3), as

$$\mathrm{AgraSSt}(\widehat{q}, t; x) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \Big| N^{-1} \sum_s \mathcal{A}^{(s)}_{\widehat{q},t} f(x) \Big|.$$

In SI.A we prove the following result for *edge-exchangeable graphs*, that is, graphs in which all permutations of the edge indicators have the same distribution.

Theorem 3.3. *If the graph is edge-exchangeable, then* $\mathrm{AgraSSt}^2(\widehat{q}, t; x)$ *is a consistent estimator of*

$$\mathrm{gKSS}^2(q; x) = N^{-2} \sum_{s, s' \in [N]} \Big\langle \mathcal{A}^{(s)}_q K(x, \cdot), \mathcal{A}^{(s')}_q K(\cdot, x) \Big\rangle_{\mathcal{H}}. \tag{9}$$

Re-sampling Stein statistic A computationally efficient operator for large N can be derived via re-sampling B vertex-pairs s_b, $b = 1, \ldots, B$, from $\{1, \ldots, N\}$, chosen uniformly with replacement, independent of each other and of x. This procedure gives rise to a randomised operator; this re-sampled operator is

$$\widehat{\mathcal{A}}^B_{\widehat{q},t} f(x) = \frac{1}{B} \sum_{b \in [B]} \mathcal{A}^{(s_b)}_{\widehat{q},t} f(x).$$

where the expectation of $\widehat{\mathcal{A}}^B_{\widehat{q},t} f(x)$ with respect to re-sampling is

$$\mathbb{E}_B[\widehat{\mathcal{A}}^B_{\widehat{q},t} f(x)] = \mathbb{E}_S[\mathcal{A}^{(S)}_{\widehat{q},t} f(x)] = \mathcal{A}_{\widehat{q},t} f(x).$$

The corresponding re-sampled AgraSSt statistic is

$$\widehat{\mathrm{AgraSSt}}(\widehat{q}, t; x) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \Big| \frac{1}{B} \sum_{b \in [B]} \mathcal{A}^{(s_b)}_{\widehat{q},t} f(x) \Big|.$$

Algorithm 2 Assessment procedures for graph generators

Input: Observed graph x; graph generator G and generated sample size L; estimation statistic t; RKHS kernel K; re-sampling size B; number of simulated graphs m; confidence level α;

Procedure:

1: Estimate $\widehat{q}(x^{(s)}|t(x_{-s}))$ based on Algorithm 1.
2: Uniformly generate re-sampling index $\{s_1, \ldots, s_B\}$ from $[N]$ with replacement.
3: Compute $\tau = \widehat{\mathrm{AgraSSt}}^2(\widehat{q}; x)$ in Eq.(10).
4: Simulate $\{z'_1, \ldots, z'_m\}$ from G.
5: Compute $\tau_i = \widehat{\mathrm{AgraSSt}}^2(\widehat{q}; z'_i)$ in Eq.(10).
6: Estimate empirical quantile $\gamma_{1-\alpha}$ via $\{\tau_1, \ldots, \tau_m\}$.

Output: Reject the null if $\tau > \gamma_{1-\alpha}$; otherwise do not reject.

Similar to Eq.(9), the squared version of $\widehat{\mathrm{AgraSSt}}$ admits a representation in a quadratic form,

$$\widehat{\mathrm{AgraSSt}}^2(\widehat{q}, t; x) = B^{-2} \sum_{b,b' \in [B]} \widehat{h}_x(s_b, s_{b'}), \tag{10}$$

where $\widehat{h}_x(s, s') = \langle \mathcal{A}^{(s)}_{\widehat{q},t} K(x, \cdot), \mathcal{A}^{(s')}_{\widehat{q},t} K(\cdot, x) \rangle_{\mathcal{H}}$. We note that the randomised operator obtained via re-sampling is a form of stochastic Stein discrepancy as introduced in Gorham et al. [2020].

For fixed x, under mild conditions the consistency of $\widehat{\mathrm{AgraSSt}}^2(\widehat{q}, t; x)$ as $B \to \infty$ is ensured by the following normal approximation, which follows from Proposition 2 in Xu and Reinert [2021].

Proposition 3.4. *Assume that $\widehat{h}_x(s, s')$ in* (10) *is bounded and that $\widehat{\mathrm{AgraSSt}}(\widehat{q}, t; x)$ has non-zero variance σ^2. Let Z be a normal variable with mean $\mathrm{AgraSSt}(\widehat{q}, t; x)$ and variance σ^2. Then there exists an explicitly computable constant $C > 0$ such that for all 3 times continuously differentiable functions g with bounded derivatives up to order 3,*

$$\mathbb{E}[g(\widehat{\mathrm{AgraSSt}}(\widehat{q}, t; x)) - g(Z)] \leq C/B.$$

4 Applications of AgraSSt

4.1 Assessing graph generators

AgraSSt measures the distributional difference between the underlying distribution of an implicit graph generator G and an observed graph x, in order to assess the quality of the generator G. The hypothesis testing procedure for the null hypothesis that the observed graph x comes from the same distribution that generates the samples, against the general alternative, is shown in Algorithm 2. We emphasise two features of this procedure. Firstly, for a given generator G, AgraSSt directly assesses the quality of the implicit model represented via samples from G. Secondly, the generator G can be trained on the observed graph x, for example through a deep neural network generator. By learning a deep neural network generator with training samples from the same distribution that generate x, AgraSSt can assess the quality of the training procedure, i.e. whether the deep neural network is capable of learning the desired distributions. Additional details are discussed in SI.C.

4.2 Interpreting trained graph generators

If the procedure in Algorithm 2 rejects the null hypothesis, the generator may not be suitable for generating samples from the distribution that generates the one observed graph. Hence, understanding where the misfit comes from can be very useful, especially for models trained from black-box deep neural networks. AgraSSt provides an interpretable model criticism by comparing the learned $\widehat{q}_t(x^{(s,1)})$ with the underlying $q_t(x^{(s,1)})$ when available, such as in synthetic experiments from a specified ERGM. Such an interpretation can be also useful to re-calibrate training procedures.

4.3 Identifying reliable graph samples

If the procedure in Algorithm 2 does not reject the null hypothesis, there is not enough evidence to reject the hypothesis that the generator is capable of generating graphs that resembles the observed

graph. If a generator G has passed this hurdle then it can be recommended for generating graph samples of the desired type. AgraSSt can also be put to use for the task of sample batch selection. In many scientific studies, only a small batch of representative graph samples may need to be generated for downstream tasks such as privacy-preserving methods where users only access a small number of graph data, or a randomised experimental design for community interaction. To quantify the quality of sample batches via p-values : (1) we generate a sample batch of size m, say; (2) we perform the steps in **Procedure** from Algorithm 2; (3) we compute the p-value $m^{-1}\sum_{i=1}^{m} \mathbb{1}(\tau > \tau_i)$, with τ as in step 3 and $\{\tau_1, \ldots, \tau_m\}$ as in step 5. If the p-value is smaller than a pre-specified threshold, we generate another sample batch; otherwise we accept the current sample batch.

5 Empirical results

We first illustrate the performance of AgraSSt on synthetic data, where the null distribution is *known* and we have control of the set-up; in particular we can illustrate the use of AgraSSt for interpretable model criticism. Then we show the performance of AgraSSt on a real-world data application to assess graph generators trained via various deep generative models.

5.1 Synthetic experiments

Only few competing approaches are available for our task and many of them are devised specifically for ERGMs. Hence here we use an ERGM, namely the Edge-2Star-Triangle (E2ST) model with

$$q(x) \propto \exp\left(\beta_1 E_d(x) + \beta_2 S_2(x) + \beta_3 T_r(x)\right), \tag{11}$$

where $E_d(x)$ denotes the number of edges of x, $S_2(x)$ denotes the 2-Star statistics and $T_r(x)$ denotes the triangle statistics. The parameter vector $\beta = (-2.00, 0.00, 0.01)$ is chosen as the null model, while alternative models are constructed by perturbing the coefficient β_2 as in Yang et al. [2018], Xu and Reinert [2021]. This particular type of ERGM is chosen because it is the currently most complex ERGM for which a thorough theoretical analysis for parameter estimation is available, see Mukherjee and Xu [2013][2].

5.1.1 Related approaches for comparisons

To assess the performance of AgraSSt, we consider the following test statistics which are either tailored or modified to perform assessment for implicit graph generators:

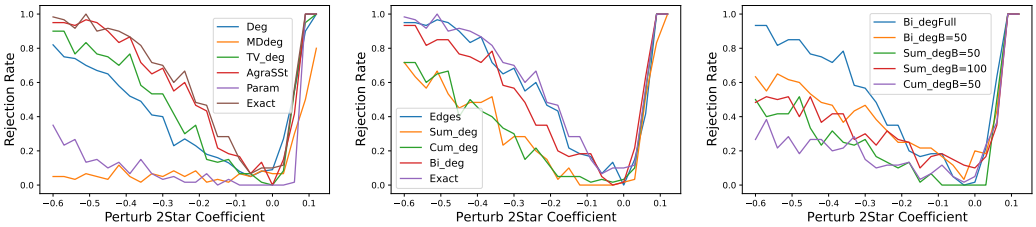
- **Deg** is a degree-based statistics for goodness-of-fit test of exchangeable random graphs [Ouadah et al., 2020] based on the estimated variance of the degree distribution. The statistics can be obtained from empirical degrees from samples generated from the implicit model.
- **TV_deg** denotes the Total-Variation (TV) distance between degree distributions. Hunter et al. [2008] proposes a simulation-based approach to construct graphical goodness-of-fit tests. Xu and Reinert [2021] quantifies this approach using the total-variation distance between the distributions of chosen network statistics; see SI.D.4 for details.
- **MDdeg** is the Mahalanobis distance between degree distributions [Lospinoso and Snijders, 2019].

In the synthetic experiment where the parametric model is known (Eq.(11)), the coefficients β are estimated from generated samples for model assessment to provide our baseline approach, denoted by **Param**. Details can be found in SI.B.1. Knowing the explicit null model in the synthetic setting, we can also compute gKSS in Eq.(3), denoted by **Exact** as our benchmark. The Weisfeiler-Lehman graph kernel [Shervashidze et al., 2011] with height parameter 3 is used for kernel-based approaches.

5.1.2 Simulation results

The rejection rates for various settings are shown in Figure.2. As the null model is relatively sparse (edge density 11.2%), the sparser alternatives are much harder to distinguish while the denser ones are

[2]The model also satisfies conditions in Theorem 1.7 in Reinert and Ross [2019] where theoretical properties are studied.



(a) Different assessment approaches (b) AgraSSt: different estimations (c) AgraSSt with re-sampling

Figure 2: Synthetic experiment on the E2ST model in Eq.(11) with β_2 varying on the x-axis: 100 trials; $\alpha = 0.05$; $L = 1000$. Abbreviations as in Section 5.1.

easier problems. Figure.2(a) compares the AgraSSt procedure with the approaches from Section 5.1.1; it shows that **AgraSSt** performs competitively to the benchmark **Exact**, which is only available when the model is known explicitly, and outperforms other assessment procedures for implicit models. **TV_deg** is slightly less powerful than **AgraSSt** but outperforms **Deg**. For sparser alternatives (smaller β_2), **MDdeg** and **Param** have a much lower rejection rate and thus are less powerful. In Figure.2(b), we compare the performance of **AgraSSt** using different estimation methods for $\widehat{q}_t(x^{(s,1)})$; the degree $deg(k)$ of a vertex k is calculated excluding the vertex-pair $s = (i, j)$. The estimation methods are

- **Sum_deg**: for $s = (i, j)$ we set $t = deg(i) + deg(j)$;
- **Cum_deg**: the cumulative distribution function of sum of degrees are used;
- **Bi_deg**: for $s = (i, j)$ the 2-dimensional vector $t = (deg(i), deg(j))$ is used;
- **Edges**: t is the edge density after removing vertex-pair s.

From the results, we see that **Edges** outperforms the other estimates, which echos the theoretical results shown in Theorem A.5 in SI, as the coefficient β for E2ST satisfies its assumptions. Hence this is the statistics which is used in our real-world experiments, with an exception of SI.D.3. We also see that using both vertex degrees as 2d vector predicts substantially better than using predictors based on sum of degrees of two vertices. In Figure.2(c), the comparison with re-sampling is shown. With increase in re-sampling size, the power of AgraSSt increases.

5.2 Real-world applications on deep graph generators

We now assess the performances of a set of state-of-the-art deep generative models for graphs trained on ERGMs and the Karate Club network collected by Zachary [1977]. The Karate Club network has 34 vertices and 78 edges representing friendships. Soon after the data collection the Karate Club separated into two factions. This graph is a benchmark graph for community detection. One would not expect this graph to be close to an $G(n, p)$[3] graph or to be well modelled by an ERGM.

5.2.1 Graph generation methods

The graph generation methods to which we apply AgraSSt are the following.

- **GraphRNN** [You et al., 2018] is an architecture to generate graphs from learning two recurrent neural networks (RNN), one a vertex-level RNN and the other an edge-level RNN. The procedure starts from a breadth-first-search for vertex ordering; two RNNs are trained from a sequential procedure.
- **NetGAN** [Bojchevski et al., 2018] utilises an adversarial approach by training an interplay between a generator and a discriminator neural network on graph data.
- **CELL** [Rendsburg et al., 2020] improves on the NetGAN idea by solving a low-rank approximation problem based on a cross-entropy objective.
- **MC** is the standard Monte-Carlo network sampling in the *ergm* suite in `R` and is used as a baseline when the simulated network is known to follow the model in Eq.(1); $q(x)$ needs to be known.

[3]Bernoulli random graph of size n, edge probability $p \in [0, 1]$.

5.2.2 Generator assessment results

We first train the generative models with samples from ERGMs to assess their ability to generate ERGMs. The test results are shown in Table.1. From the result, we see that for the "reliable" **MC** generator all the assessment statistics presented have well-controlled type-I error. Samples generated from **CELL** deviate not too far from the test level, indicating a good generative model for ERGMs. **NetGAN** and **GraphRNN** both encounter a high rejection rate, implying that the generated samples that are not close to the training E2ST model.

	AgraSSt	Deg	MDdeg	TV_deg
GraphRNN	0.42	0.02	0.04	0.27
NetGAN	0.81	0.13	0.61	0.54
CELL	0.05	0.06	0.09	0.12
MC	0.04	0.03	0.02	0.09

Table 1: Rejection rates on various assessment approaches, with $L = 1000$; 200 samples to simulate the null; 100 trials; $\alpha = 0.05$. The higher the rejection rate, the worse the model fit. MC is the baseline.

From the density based **AgraSSt**, taking a $G(n, q)$ model, we can interpret the model misfit by checking the estimated $\widehat{q}$. For the true E2ST model used to generate training samples, we have $= 0.112$, while CELL has $\widehat{q} = 0.116$ which is close to the null. GraphRNN estimates $\widehat{q} = 0.128$ which is substantially higher than the null. Although GraphRNN can be powerful in learning local patterns and structures for neighbourhoods [You et al., 2018], here it does not take the overall density sufficiently into account. Due to its limited "look back" and absence of "look forward" on the ordered vertex set during training, the over-generation of edges may have caused this significant difference for learning ERGMs. NetGAN, on the other hand, produces a close estimate $\widehat{q} = 0.106$. However, counting triangles, it only has on average 12.6 triangles , which is far less from the null with expected number of triangles 46.3. NetGAN, due to its random walk adversarial procedure, may not be effective in learning such clustered patterns.

5.2.3 Case study: Karate Club network

Next, we assess the performances of these generative models by training on the Karate Club network [Zachary, 1977]. The p-values for different testing procedures are shown in Table.2. From the results, we see that AgraSSt rejects samples generated from both GraphRNN and NetGAN trained with the Karate Club network. Although the edge densities generated from the trained GraphRNN (edge density 15.3%) and NetGAN (edge density 13.4%) are comparable with the Karate Club edge density of 13.9%, visual inspection indicates that both GraphRNN and NetGAN samples exhibit a single large component rather than two fairly separated communities in the Karate Club network. This difference is picked up by AgraSSt, Deg and TV_deg, which all reject both models. On the other hand, CELL generates samples that are not rejected by all tests at significance level $\alpha = 0.05$. In Figure.5 in the SI, the Karate Club network is shown in Figure.5(a). Samples from GraphRNN, NetGAN and CELL are shown in Figure.5(b), 5(c) and 5(d) respectively.

	AgraSSt	Deg	MDdeg	TV_deg
GraphRNN	0.00	0.01	0.15	0.00
NetGAN	0.00	0.02	0.59	0.00
CELL	0.34	0.09	0.17	0.61

Table 2: p-values for models trained on the Karate Club network; 100 samples to simulate the null distribution; rejection at $\alpha = 0.05$ is marked red.

SI D includes additional results, including on efficiency, and visualisations. A second case study — the Florentine marriage network from Padgett and Ansell [1993] — is presented in SI D.2; additional visualisations of the reliable sample batch selection procedure described in Section 4.3 are also given.

6 Discussions and future directions

In this paper, we propose AgraSSt, a unique general purpose model assessment and criticism procedure for implicit random graph models. As it is based on a kernel Stein statistic, we are able to give theoretical guarantees. AgraSSt not only solves an important problem but also opens up a whole set of follow-up research problems of which we list a few here. (i). Currently AgraSSt is only applied to undirected and unweighted graphs. Extensions to more general graphs as well as to time

series of graphs will be interesting to explore in follow-up work. (ii). AgraSSt could be also helpful to improve design and training of deep graph generative models, e.g. by regularising graph features if there is a misfit. (iii). ERGMs allow for exogenous features to be included in the sufficient statistics. AgraSSt can be based on a variety of statistics $t(x)$; further examples are found in SI D.3. It would also be possible to incorporate exogenous features in the statistics $t(x)$ in AgraSSt, for example using ideas from graph attention networks [Veličković et al., 2018]. Exploring this idea in more detail will be another topic of further research.

As AgraSSt depends on the chosen summary statistic $t(x)$, results have to be interpreted with regards to the respective conditional distributions. Also multiple tests will have the p-values to be adjusted to avoid misinterpretation of tests, which could have serious consequences for example in the area of personal health. Investigating the effect of choice of $t(x)$ will be part of future work.

Acknowledgements

The authors thank the anonymous reviewers for their constructive comments and suggestions to improve the paper. G.R. and W.X. acknowledges the support from EPSRC grant EP/T018445/1. G.R is also supported in part by EPSRC grants EP/W037211/1, EP/V056883/1, and EP/R018472/1.

References

Alessandro Barp, Francois-Xavier Briol, Andrew Duncan, Mark Girolami, and Lester Mackey. Minimum Stein discrepancy estimators. In *Advances in Neural Information Processing Systems*, pages 12964–12976, 2019.

Alain Berlinet and Christine Thomas. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2004.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 24(3):179–195, 1975.

Shankar Bhamidi, Guy Bresler, and Allan Sly. Mixing time of exponential random graphs. *The Annals of Applied Probability*, 21(6):2146–2170, 2011.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *International Conference on Machine Learning*, pages 610–619. PMLR, 2018.

Guy Bresler and Dheeraj Nagaraj. Optimal single sample tests for structured versus unstructured network data. In *Conference On Learning Theory*, pages 1657–1690. PMLR, 2018.

Guy Bresler and Dheeraj Nagaraj. Stein's method for stationary distributions of Markov chains and application to Ising models. *The Annals of Applied Probability*, 29(5):3230–3265, 2019.

Sourav Chatterjee and Persi Diaconis. Estimating and understanding exponential random graph models. *The Annals of Statistics*, 41(5):2428–2461, 2013.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *International Conference on Machine Learning*, pages 2606–2615. PMLR, 2016.

Kacper P Chwialkowski, Dino Sejdinovic, and Arthur Gretton. A wild bootstrap for degenerate kernel tests. In *Advances in Neural Information Processing Systems*, pages 3608–3616, 2014.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International Conference on Machine Learning*, pages 2302–2312. PMLR, 2020.

Ronen Eldan and Renan Gross. Exponential random graphs behave like mixtures of stochastic block models. *The Annals of Applied Probability*, 28(6):3698–3735, 2018.

Ove Frank and David Strauss. Markov graphs. *Journal of the American Statistical Association*, 81 (395):832–842, 1986.

Jackson Gorham and Lester Mackey. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234, 2015.

Jackson Gorham, Anant Raj, and Lester Mackey. Stochastic Stein discrepancies. *Advances in Neural Information Processing Systems*, 33:17931–17942, 2020.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pages 513–520, 2007.

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 *Advances in Neural Information Processing Systems*, pages 1205–1213, 2012.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Paul W Holland and Samuel Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76(373):33–50, 1981.

David R Hunter and Mark S Handcock. Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, 15(3):565–583, 2006.

David R Hunter, Steven M Goodreau, and Mark S Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481):248–258, 2008.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

Aapo Hyvärinen. Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, 18(10):2283–2292, 2006.

Bai Jiang, Tung-Yu Wu, Yifan Jin, and Wing H Wong. Convergence of contrastive divergence algorithm in exponential family. *The Annals of Statistics*, 46(6A):3067–3098, 2018.

Wittawat Jitkrittum, Zoltán Szabó, Kacper P Chwialkowski, and Arthur Gretton. Interpretable distribution features with maximum testing power. In *Advances in Neural Information Processing Systems*, pages 181–189, 2016.

Wittawat Jitkrittum, Wenkai Xu, Zoltán Szabó, Kenji Fukumizu, and Arthur Gretton. A linear-time kernel goodness-of-fit test. In *Advances in Neural Information Processing Systems*, pages 262–271, 2017.

Christophe Ley, Gesine Reinert, and Yvik Swan. Stein's method for comparison of univariate distributions. *Probability Surveys*, 14:1–52, 2017.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

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. *Advances in Neural Information Processing Systems*, 32:4255–4265, 2019.

Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Danica J Sutherland. Learning deep kernels for non-parametric two-sample tests. In *International Conference on Machine Learning*, pages 6316–6326. PMLR, 2020.

Feng Liu, Wenkai Xu, Jie Lu, and Danica J Sutherland. Meta two-sample testing: Learning kernels for testing with limited data. *Advances in Neural Information Processing Systems*, 34, 2021.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning*, pages 276–284, 2016.

Josh Lospinoso and Tom AB Snijders. Goodness of fit for stochastic actor-oriented models. *Methodological Innovations*, 12(3):2059799119884282, 2019.

Guillaume Mijoule, Gesine Reinert, and Yvik Swan. Stein's density method for multivariate continuous distributions. *arXiv preprint arXiv:2101.05079*, 2021.

Sumit Mukherjee and Yuanzhe Xu. Statistics of the two-star ERGM. *arXiv preprint arXiv:1310.4526*, 2013.

Sarah Ouadah, Stéphane Robin, and Pierre Latouche. Degree-based goodness-of-fit tests for heterogeneous random graph models: Independent and exchangeable cases. *Scandinavian Journal of Statistics*, 47(1):156–181, 2020.

John F Padgett and Christopher K Ansell. Robust Action and the Rise of the Medici, 1400-1434. *American Journal of Sociology*, 98(6):1259–1319, 1993.

Gesine Reinert and Nathan Ross. Approximating stationary distributions of fast mixing Glauber dynamics, with applications to exponential random graphs. *The Annals of Applied Probability*, 29 (5):3201–3229, 2019.

Luca Rendsburg, Holger Heidrich, and Ulrike Von Luxburg. NetGAN without GAN: From random walks to low-rank approximations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8073–8082. PMLR, 2020.

Christian S Schmid and Bruce A Desmarais. Exponential random graph models with big networks: Maximum pseudolikelihood estimation and the parametric bootstrap. In *2017 IEEE International Conference on Big Data*, pages 116–121. IEEE, 2017.

Cosma Rohilla Shalizi and Alessandro Rinaldo. Consistency under sampling of exponential random graph models. *Annals of Statistics*, 41(2):508, 2013.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.

Tom AB Snijders. Markov chain Monte Carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2):1–40, 2002.

David Strauss and Michael Ikeda. Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, 85(409):204–212, 1990.

Marijtje AJ Van Duijn, Krista J Gile, and Mark S Handcock. A framework for the comparison of maximum pseudo-likelihood and maximum likelihood estimation of exponential family random graph models. *Social Networks*, 31(1):52–62, 2009.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*, volume 8. Cambridge University Press, 1994.

Wenkai Xu and Takeru Matsuda. A Stein goodness-of-fit test for directional distributions. *International Conference on Artificial Intelligence and Statistics*, 2020.

Wenkai Xu and Takeru Matsuda. Interpretable Stein goodness-of-fit tests on Riemannian manifolds. *International Conference on Machine Learning*, 2021.

Wenkai Xu and Gesine Reinert. A Stein goodness-of-test for exponential random graph models. In *International Conference on Artificial Intelligence and Statistics*, pages 415–423. PMLR, 2021.

Jiasen Yang, Qiang Liu, Vinayak Rao, and Jennifer Neville. Goodness-of-fit testing for discrete distributions via Stein discrepancy. In *International Conference on Machine Learning*, pages 5557–5566, 2018.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

Checklist

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] SI.A in Supplementary
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code attached as a folder in the Supplementary
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Model details are included in the code. Additional implementation details are included in SID.6.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Random seed are used for experiment and included in the code. See SID.3 for error bars for graphical tests.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See SID.5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5
   (b) Did you mention the license of the assets? [N/A] Datasets and codes are open source.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] SID
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs and additional theoretical results

A.1 Proofs

Here we usually omit the $= \underline{k}$ suffix in $\mathcal{A}^{(s)}_{q,t=\underline{k}}$ and write instead $\mathcal{A}^{(s)}_{q,t}$ for easier reading. The operator $\mathcal{A}^{(s)}_{q,t}$ is understood as having a fixed value $t(x_s)$ of t observed. Similarly we often write $\mathbb{E}_{q,t}$ to abbreviate $\mathbb{E}_{q,t=\underline{k}}$, with the same understanding.

Proof of Lemma 3.1 For convenience we repeat the statement of the lemma here.

Lemma 3.1 *If $q_{\underline{k}}(x) > 0$ for all $\underline{k}$, then $\mathcal{A}^{(s)}_{q,t=\underline{k}}$ is a Stein operator for the conditional distribution of X given $t(X) = \underline{k}$, and $\sum_s \mathcal{A}^{(s)}_{q,t=\underline{k}}$ is a Stein operator for the conditional distribution of X given $t(X) = \underline{k}$.*

Proof. In order to show the assertion we prove that for $\mathbb{E}_{q,t=\underline{k}}$ denoting the conditional distribution of X given $t(X) = \underline{k}$, the expectation $\mathbb{E}_{q,t=\underline{k}}[\mathcal{A}^{(s)}_{q,t} f]$ vanishes for all functions for which the expectation exists. Let f be such a function. We have

$$\begin{aligned}
\mathcal{A}^{(s)}_{q,t} f(x^{(s,1)}) =& q(x^{(s,0)}|t(x_s)= \underline{k})(f(x^{(s,0)}) - f(x^{(s,1)}))\\
\mathcal{A}^{(s)}_{q,t} f(x^{(s,0)}) =& q(x^{(s,1)}|t(x_s)= \underline{k})(f(x^{(s,1)}) - f(x^{(s,0)})).
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathbb{E}_{q,t}[\mathcal{A}^{(s)}_{q,t} f] =& \sum_{x_{-s}} \Big\{ \mathbb{1}(x_s = 1) p_t(x^{(s,1)}) q(y^{(s,0)}|t(y_s)= \underline{k})(f(x^{(s,0)}) - f(x^{(s,1)}))\\
& -\mathbb{1}(x_s = 0) p_t(x^{(s,0)}) q(y^{(s,1)}|t(y_s)= \underline{k})(f(x^{(s,0)}) - f(x^{(s,1)}))\Big\}\\
=& \sum_{x_{-s}} (f(x^{(s,0)}) - f(x^{(s,1)})) p_t(x^{(s,1)}) p_t(x^{(s,0)}) \left\{\mathbb{1}(x_s = 1) - \mathbb{1}(x_s = 0)\right\}\\
=& 0.
\end{aligned}$$

□

Proof of Theorem 3.2 For convenience we repeat the theorem here.

Theorem 3.2 *Assume that $\widehat{q}_t(x^{(s,1)})$ is a consistent estimator for $q_t(x^{(s,1)})$ as $L \to \infty$. Then for any function f such that $||\Delta f|| < \infty$ we have $\mathbb{E}_q[\mathcal{A}_{\widehat{q},t} f(x)] \to \mathbb{E}_q[\mathcal{A}_{q,t} f(x)] = 0$ as $L \to \infty$.*

Proof. We recall the notation that Equation (4). We have that

$$\mathcal{A}_{q,t} f(x) = \frac{1}{N} \sum_{s\in[N]} [q(x^{(s,1)}|t(x_{-s}) f(x^{(s,1)}) + q(x^{(s,0)}|t(x_{-s}) f(x^{(s,0)}) - f(x)]$$

so that

$$\begin{aligned}
\mathcal{A}_{\widehat{q},t} f(x) - \mathcal{A}_{q,t} f(x) =& \frac{1}{N} \sum_{s\in[N]} \{(\widehat{q}(x^{(s,1)}|t(x_{-s})) - q(x^{(s,1)}|t(x_{-s})) f(x^{(s,1)})\\
& + (1 - \widehat{q}(x^{(s,1)}|t(x_{-s}) - (1 - q(x^{(s,1)}|t(x_{-s}))) f(x^{(s,0)})\}\\
=& \frac{1}{N} \sum_{s\in[N]} (\widehat{q}(x^{(s,1)}|t(x_{-s}) - q(x^{(s,1)}|t(x_{-s})) \Delta_s f(x).
\end{aligned}$$

Hence

$$|\mathcal{A}_{\widehat{q},t} f(x) - \mathcal{A}_{q,t} f(x)| \leq ||\Delta f|| \frac{1}{N} \sum_{s\in[N]} |\widehat{q}(x^{(s,1)}|t(x_{-s})) - q(x^{(s,1)}|t(x_{-s}))|.$$

Thus, if for all $s \in [N]$, as $L \to \infty$ we have $\widehat{q}(x^{(s,1)}|t(x_{-s})) - q(x^{(s,1)}|t(x_{-s})) \to 0$ for all s then so does $|\mathcal{A}_{\widehat{q},t} f(x) - \mathcal{A}_{q,t} f(x)|$. The assertion follows from the assumption that $\widehat{q}_t(x^{(s,1)})$ is a consistent estimator for $q_t(x^{(s,1)})$ as $L \to \infty$. □

Proof of Theorem 3.3 For convenience we repeat the statement of the theorem here. Recall that a random graph model is *edge-exchangeable* if its edge indicator variables are finitely exchangeable. Often we just write *edge-exchangeable graph*.

Theorem 3.3 *If the graph is edge-exchangeable, then* $\mathrm{AgraSSt}^2(\widehat{q}, t; x)$ *is a consistent estimator of*

$$\mathrm{gKSS}^2(q; x) = N^{-2} \sum_{s,s' \in [N]} \left\langle \mathcal{A}_q^{(s)} K(x, \cdot), \mathcal{A}_q^{(s')} K(\cdot, x) \right\rangle_{\mathcal{H}}.$$

For easier tractability the proof is organised in two steps.

1. First, Proposition A.1 shows that in an edge-exchangeable random graph model, $g_{\underline{k}}$ given in Equation (8) is a consistent estimator for $q(x^{(s,1)}|t(x_{-s}))$ as $NL \to \infty$.
2. Theorem A.2 uses these results to obtain a concentration bound for $\mathcal{A}_{\widehat{q}}$ from which then Theorem 3.3 follows.

Moreover theoretical guarantees for fixed L which depend on the model are given. As the graph generator can generate as large a number L of graphs as desired, these theoretical results can be used to determine L which result in theoretical guarantees on deviations from the mean.

Proposition A.1. *Suppose that $X_1, \ldots, X_L$ are i.i.d. copies of the adjacency matrix of an edge-exchangeable random graph model. Let $s = (i, j)$ be a fixed vertex-pair. For $l = 1, \ldots, L$ and for a graph X_l let $t_l^{(s)}$ denote a possibly multivariate statistic which is evaluated on the collection of indicator variables in X_l except $X_{s,l}$. For a possible $t_l^{(s)}$ outcome $\underline{k}$, let $p(\underline{k}) = \mathbb{P}(t_l^{(s)} = \underline{k})$ and let $\underline{k}$ be such that $p(\underline{k}) \neq 0$. Set*

$$p(1; \underline{k}) = \mathbb{P}(X_s = 1 | t^{(s)} = \underline{k});$$

let

$$n(\underline{k}, s) = \sum_{l=1}^{L} X_{s,l} \mathbb{1}(t_l^{(s)} = \underline{k}) \quad \textit{and} \quad n(\underline{k}^{(s)}) = \sum_{l=1}^{L} \mathbb{1}(t_l^{(s)} = \underline{k});$$

$$n(\underline{k}) = \sum_{s \in [N]} n(\underline{k}, s) \quad \textit{and} \quad N_{\underline{k}} = \sum_{s \in [N]} n(\underline{k}^{(s)});$$

and set

$$g(\underline{k}) = \frac{n(\underline{k})}{N_{\underline{k}}} \mathbb{1}(N_{\underline{k}} \geq 1).$$

Then $g(\underline{k}) \to p$ in probability as $NL \to \infty$. In particular, for all $\epsilon > 0$,

$$\mathbb{P}\left[|\hat{g}(\underline{k}) - p(1; \underline{k})| > \epsilon\right] \leq \frac{4}{\epsilon^2 NL} \{p(s, \underline{k})[1 - p(s, \underline{k}) p(\underline{k})] + 1 - p(\underline{k})\}.$$

Proof. Due to the exchangeability of the edges we have, with s denoting a generic edge,

$$\mathbb{E}(n(\underline{k})) = N\mathbb{E}(n(\underline{k}, s)) = NLp(1; \underline{k})p(\underline{k}), \qquad Var(n(\underline{k})) = NLp(1; \underline{k})p(\underline{k})(1 - p(1; \underline{k})p(\underline{k})),$$

as well as

$$\mathbb{E}(N_{\underline{k}}) = N\mathbb{E}(n(\underline{k}^{(s)})) = NLp(\underline{k}), \qquad Var(N_{\underline{k}}) = NLp(\underline{k})(1 - p(\underline{k})).$$

To show convergence in probability, let $\epsilon > 0$. Then

$$\begin{aligned}\mathbb{P}\left[|\hat{g}(\underline{k}) - p(1, \underline{k})| > \epsilon\right] \leq & \mathbb{P}\left[\left|\frac{n(\underline{k})}{\mathbb{E}(n(\underline{k}))} \mathbb{1}[n(\underline{k}) \geq 1] - p(1, \underline{k})\right| > \frac{1}{2}\epsilon\right] \\ & + \mathbb{P}\left[\left|n(\underline{k}) \mathbb{1}[n(\underline{k}) \geq 1] \left(\frac{1}{n(\underline{k})} - \frac{1}{\mathbb{E}(n(\underline{k}))}\right)\right| > \frac{1}{2}\epsilon\right].\end{aligned}$$

Note that $n(\underline{k})\mathbb{1}[n(\underline{k}) \geq 1] = n(\underline{k})$. By Chebychev's inequality,

$$\begin{aligned}\mathbb{P}\left[\left|\frac{n(\underline{k})}{\mathbb{E}(N_{\underline{k}})} \mathbb{1}[N_{\underline{k}} \geq 1] - p(1; \underline{k})\right| > \frac{1}{2}\epsilon\right] &\leq \frac{4}{\epsilon^2 N^2 L^2 p(\underline{k})^2} NLp(1; \underline{k})p(\underline{k})(1 - p(1; \underline{k})p(\underline{k})) \\ &= \frac{4}{\epsilon^2 NLp(\underline{k})} p(1; \underline{k})(1 - p(1; \underline{k})p(\underline{k}))\end{aligned}$$

and

$$\mathbb{P}\left[\left|n(\underline{k})\mathbb{1}[N_{\underline{k}} \geq 1]\left(\frac{1}{n(\underline{k})} - \frac{1}{\mathbb{E}(N_{\underline{k}})}\right)\right| > \frac{1}{2}\epsilon\right] \leq \mathbb{P}\left[\frac{1}{\mathbb{E}(N_{\underline{k}})}\left|\mathbb{E}(N_{\underline{k}}) - N_{\underline{k}}\right| > \frac{1}{2}\epsilon\right]$$
$$\leq \frac{4}{\epsilon^2 NLp(\underline{k})}(1 - p(\underline{k})).$$

Summing the contributions completes the proof. □

Proposition A.1 shows that in edge-exchangeable graphs, $\hat{g}(\underline{k})$ consistently estimates $q(x^{(s,1)}) = q(x^{(s,1)}|t(x_{-s}) = \underline{k})$. In an expanded version of Theorem 3.3 we show that the approximate Stein operator from Eq.(7),

$$\mathcal{A}_{\widehat{q},t}f(x) := \frac{1}{N}\sum_{s\in[N]} \mathcal{A}_{\widehat{q}(x^{(s)}|t(x_{-s}))}f(x),$$

with

$$\widehat{q}(x^{(s,1)}|t(x_{-s})) = g(x_{-s}), \quad \widehat{q}(x^{(s,0)}|t(x_{-s})) = 1 - g(x_{-s})$$

is a consistent estimator of

$$\mathcal{A}_{q,t}f(x) := \frac{1}{N}\sum_{s\in[N]} \mathcal{A}_{q(x^{(s)}|t(x_{-s}))}f(x).$$

We recall

$$\mathrm{AgraSSt}^2(\hat{q}, t, x) = \frac{1}{N^2}\sum_{s,s'\in[N]} h_x(s, s')$$

with

$$h_x(s, s') = \left\langle \mathcal{A}^{(s)}_{\hat{q},t}K(x,\cdot), \mathcal{A}^{(s')}_{\hat{q},t}K(\cdot, x)\right\rangle_{\mathcal{H}}.$$

We state the expanded version of Theorem 3.3 here.

Theorem A.2. *If the graph is edge-exchangeable then for any test function f for which the Stein operator $\mathcal{A}_{q,t}f$ is well defined, and for all $\epsilon > 0$*

$$\mathbb{P}(|\mathcal{A}_{\widehat{q},t}f(X) - \mathcal{A}_{q,t}f(X)| > \epsilon) \leq \frac{4}{\epsilon^2 NL(||\Delta f||)^{-2}}\left(\{p(1,\underline{k})[1 - p(1,\underline{k})p(\underline{k})] + 1 - p(\underline{k})\}\right).$$

Moreover, $\mathrm{AgraSSt}^2(\hat{q}, t, x)$ *is a consistent estimator of*

$$\mathrm{gKSS}(x) = \frac{1}{N^2}\sum_{s,s'\in[N]} \left\langle \mathcal{A}^{(s)}_{q,t}K(x,\cdot), \mathcal{A}^{(s')}_{q,t}K(\cdot, x)\right\rangle_{\mathcal{H}}.$$

Proof. We have that

$$\mathcal{A}_{q,t}f(x) := \frac{1}{N}\sum_{s\in[N]} \mathcal{A}_{q(x^{(s)}|t(x_{-s}))}f(x).$$

and

$$\mathcal{A}_{\widehat{q},t}f(x) := \frac{1}{N}\sum_{s\in[N]} \mathcal{A}_{\widehat{q}(x^{(s)}|t(x_{-s}))}f(x),$$

with

$$\widehat{q}(x^{(s,1)}|t(x_{-s})) = g(x_{-s}), \quad \widehat{q}(x^{(s,0)}|t(x_{-s})) = 1 - g(x_{-s})$$

so that

$$\begin{aligned}\mathcal{A}_{\widehat{q},t}f(x) - \mathcal{A}_{q,t}f(x) &= \frac{1}{N}\sum_{s\in[N]}\{(g(\underline{k}) - q(x^{(s)}|t(x_{-s})))f(x^{(s,1)}) \\ &\qquad + (1 - g(\underline{k}) - (1 - q(x^{(s)}|t(x_{-s}))))f(x^{(s,0)})\} \\ &= \frac{1}{N}\sum_{s\in[N]}(g(\underline{k}) - q(x^{(s)}|t(x_{-s})))\{f(x^{(s,1)}) - f(x^{(s,0)})\} \\ &= \frac{1}{N}\sum_{s\in[N]}(g(\underline{k}) - q(x^{(s)}|t(x_{-s})))\Delta_s f(x).\end{aligned}$$

Hence

$$|\mathcal{A}_{\widehat{q},t}f(x) - \mathcal{A}_{q,t}f(x)| \leq ||\Delta f|| \frac{1}{N} \sum_{s\in[N]} |g(\underline{k}) - q(x^{(s)}|t(x_{-s}))|.$$

With Proposition A.1 and using the edge-exchangeability,

$$\mathbb{P}(|\mathcal{A}_{\widehat{q},t}f(X) - \mathcal{A}_{q,,t}f(X)| > \epsilon) \leq \frac{4}{\epsilon^2 NL(||\Delta f||)^{-2}} \left(\{p(1,\underline{k})[1-p(1,\underline{k})p(\underline{k})] + 1 - p(\underline{k})\}\right).$$

The fact that taking the sup over functions in the Hilbert space $\mathcal{H}$ does not spoil the convergence follows from the closed form representation of the sup of $\mathrm{AgraSSt}^2$, see for example Equation (11) in [Xu and Reinert, 2021]. We have that

$$\mathrm{AgraSSt}^2(\hat{q},t,x) = \frac{1}{N^2} \sum_{s,s'\in[N]} h_x(s,s')$$

where

$$h_x(s,s') = \left\langle \mathcal{A}^{(s)}_{\hat{q},t} K(x,\cdot), \mathcal{A}^{(s')}_{\hat{q},t} K(\cdot,x) \right\rangle_{\mathcal{H}}.$$

Hence,

$$\begin{aligned} &\mathrm{AgraSSt}^2(\widehat{q}) - \mathrm{AgraSSt}^2(q) \\ &= \frac{1}{N^2} \sum_{s,s'\in[N]} \left\langle \mathcal{A}^{(s)}_{\widehat{q},t} K(x,\cdot) - \mathcal{A}^{(s)}_{q,t} K(x,\cdot), \mathcal{A}^{(s')}_{\widehat{q},t} K(\cdot,x) - \mathcal{A}^{(s)}_{q,t} K(x,\cdot) \right\rangle_{\mathcal{H}} \end{aligned}$$

and the first part gives the desired convergence as $L \to \infty$. □

A.2 Gaussian approximation for AgraSSt in ERGMs

If the observed graph x is a realisation of an ERGM then results from Xu and Reinert [2021] can be leveraged to obtain finer theoretical results. First we detail the scaling for exponential random graph models which is used in the theoretical results which follow. For a graph H on at most n vertices $V(H)$ denote the vertex set, and for $x \in \{0,1\}^N$, denote by $t(H,x)$ the number of *edge-preserving* injections from $V(H)$ to $V(x)$; an injection σ preserves edges if for all edges vw of H with $\sigma(v) < \sigma(w)$, $x_{\sigma(v)\sigma(w)} = 1$. For $v_H = |V(H)| \geq 3$ set

$$t_H(x) = \frac{t(H,x)}{n(n-1)\cdots(n-v_H+3)}.$$

If $H = H_1$ is a single edge, then $t_H(x)$ is twice the number of edges of x. In the exponent this scaling of counts matches Definition 1 in Bhamidi et al. [2011] and Sections 3 and 4 of Chatterjee and Diaconis [2013]. An ERGM for the collection $x \in \{0,1\}^N$ can be defined as follows.

Definition A.3 (Definition 1.5 in Reinert and Ross [2019])**.** Fix $n \in \mathbb{N}$ and $k \in \mathbb{N}$. Let H_1 be a single edge and for $l = 2,\ldots,k$ let H_l be a connected graph on at most n vertices; set $t_l(x) = t_{H_l}(x)$. For $\beta = (\beta_1,\ldots,\beta_k)^\top \in \mathbb{R}^k$ and $t(x) = (t_1(x),\ldots,t_k(x))^\top \in \mathbb{R}^k$ $X \in \mathcal{G}_n^{lab}$ follows the exponential random graph model $X \sim \mathrm{ERGM}(\beta,t)$ if for $\forall x \in \mathcal{G}_n^{lab}$,

$$q(X=x) = \frac{1}{\kappa_n(\beta)} \exp\left(\sum_{l=1}^k \beta_l t_l(x)\right).$$

Here $\kappa_n(\beta)$ is a normalisation constant.

In particular, under suitable conditions, the ERGM Glauber Stein operator is close to the $G(n,p)$ Stein operator. This result is already shown in Reinert and Ross [2019], Theorem 1.7, with details provided in the proof of Theorem 1 in Xu and Reinert [2021]. To give the result, a technical assumption is required, which originates in Chatterjee and Diaconis [2013], and is required in Reinert and Ross [2019]. For $a \in [0,1]$, define the following functions [Bhamidi et al., 2011, Eldan and Gross, 2018], with the notation in Definition A.3 for ERGM(β,t):

$$\Phi(a) := \sum_{l=1}^k \beta_l e_l a^{e_l - 1}, \quad \varphi(a) := \frac{1+\tanh(\Phi(a))}{2}$$

where e_l is the number of edges in H_l.

Assumption 1. (1) $\frac{1}{2}|\Phi|'(1) < 1$. (2) $\exists a^* \in [0,1]$ that solves the equation $\varphi(a^*) = a^*$.

The value a^* will be the edge probability in the approximating Bernoulli random graph, $\mathrm{ER}(a^*)$. The following result holds.

Proposition A.4. *Let $q(x) = \mathrm{ERGM}(\beta, t)$ satisfy Assumption 1 and let $\tilde{q}$ denote the distribution of ER(a^*). Then there is an explicit constant $C = C(\beta, t, K)$ such that for all $\epsilon > 0$,*

$$\frac{1}{N}\sum_{s\in N}\mathbb{E}|(\mathcal{A}_q^{(s)}f(Y) - \mathcal{A}_{\tilde{q}}^{(s)}f(Y))| \leq ||\Delta f||\binom{n}{2}\frac{C(\beta,t)}{\sqrt{n}}.$$

Moreover, for $f \in \mathcal{H}$ equipped with kernel K, let $f_x^*(\cdot) = \frac{(\mathcal{A}_q - \mathcal{A}_{\tilde{q}})K(x,\cdot)}{\|(\mathcal{A}_q - \mathcal{A}_{\tilde{q}})K(x,\cdot)\|_{\mathcal{H}}}$. Then there is an explicit constant $C = C(\beta, t, K)$ such that for all $\epsilon > 0$,

$$\mathbb{P}(|\mathrm{gKSS}(q, X) - \mathrm{gKSS}(\tilde{q}, Y)| > \epsilon) \tag{12}$$

$$\leq \left\{||\Delta(\mathrm{gKSS}(q,\cdot))^2||(1 + ||\Delta\mathrm{gKSS}(q,\cdot)||) + 4\sup_x(||\Delta f_x^*||^2)\right\}\binom{n}{2}\frac{C}{\epsilon^2\sqrt{n}}. \tag{13}$$

Proof. The assertion follows immediately from the proof of Theorem 1 in Xu and Reinert [2021]. □

The approximation with a Bernoulli random graph is useful as for a Bernoulli random graphs a normal approximation for its gKSS is available in Xu and Reinert [2021], under suitable assumptions.

Assumption 2. Let $\mathcal{H}$ be the RKHS associated with the kernel $K : \{0,1\}^N \times \{0,1\}^N \to \mathbb{R}$ and for $s \in [N]$ let $\mathcal{H}_s$ be the RKHS associated with the kernel $l_s : \{0,1\} \times \{0,1\} \to \mathbb{R}$. Then

i) $\mathcal{H}$ is a tensor product RKHS, $\mathcal{H} = \otimes_{s\in[n]}\mathcal{H}_s$;
ii) k is a product kernel, $k(x,y) = \otimes_{s\in[N]} l_s(x_s, y_s)$;
iii) $\langle l_s(x_s,\cdot), l_s(x_s,\cdot)\rangle_{\mathcal{H}_s} = 1$;
iv) $l_s(1,\cdot) - l_s(0,\cdot) \neq 0$ for all $s \in [N]$.

These assumptions are satisfied for example for the suitably standardised Gaussian kernel $K(x,y) = \exp\{-\frac{1}{\sigma^2}\sum_{s\in[N]}(x_s - y_s)^2\}$.

Letting $||\cdot||_1$ denote L_1-distance, and $\mathcal{L}$ denote the law of a random variable, Xu and Reinert [2021] show the following normal approximation.

Theorem A.5 (Theorem 2 in Xu and Reinert [2021])**.** *Let Y have the distribution $\tilde{q}$ of a Bernoulli random graph $ER(a^*)$ as in Proposition A.4. Assume that the conditions i) - iv) in Assumption 2 hold. Let $\mu = \mathbb{E}[\mathrm{gKSS}^2(\tilde{q}, Y)]$ and $\sigma^2 = Var[\mathrm{gKSS}^2(\tilde{q}, Y)]$. Set $W = \frac{1}{\sigma}(\mathrm{gKSS}^2(\tilde{q}, Y)] - \mu)$ and let Z denote a standard normal variable, Then there is an explicit constant $C = C(a^*, l_s, s \in [N])$ such that*

$$||\mathcal{L}(W) - \mathcal{L}(Z)||_1 \leq \frac{C}{\sqrt{N}}.$$

Thus a normal approximation for the approximating gKSS can then be used to assess the theoretical behaviour of AgraSSt as follows.

Corollary A.6. *Let the assumptions Proposition A.4 and Theorem A.5 be satisfied. With the notation of Theorem A.5, assume that the RKHS kernel K is such that the right hand side of Proposition A.4 is $o(n)$. Then $\frac{1}{\sigma}(\mathrm{AgraSSt}(\widetilde{q}(x^{(s)}|t(x_{-s}))) - \mu)$ is approximately standard normally distributed as $N \to \infty$.*

Proof. For all $\epsilon > 0$,

$$\mathbb{P}\left[\left|\mathrm{AgraSSt}(\widetilde{q}(x^{(s)}|t(x_{-s}))) - \mathrm{gKSS}(a^*)\right| > \epsilon\right]$$
$$\leq \mathbb{P}\left[\left|\mathrm{AgraSSt}(\widetilde{q}(x^{(s)}|t(x_{-s}))) - \mathrm{gKSS}(q)\right| > \frac{1}{2}\epsilon\right] + \mathbb{P}\left[|\mathrm{gKSS}(q) - \mathrm{gKSS}(a^*)| > \frac{1}{2}\epsilon\right].$$

The first summand tends to 0 as $N \to \infty$ due to Theorem 3.2 and the second summand tends to 0 due to Proposition A.4. That $\mathrm{gKSS}(a^*)$ is approximately normally distributed with the appropriate scaling follows from Theorem A.5. □

The theoretical behaviour of the subsampling version $\widehat{\mathrm{AgraSSt}}(\widetilde{q}(x^{(s)}|t(x_{-s})))$ is addressed in Proposition 3.4. A detailed examination of the choice of kernel K such that the assumptions of Corollary A.6 are satisfied is left for future work.

B Additional background

In this section, we present additional background to complement the discussions in the main text.

B.1 Parameter estimation for random graphs

Estimating parameters for parametric models is possible *only* when the parametric family is *explicitly specified*. For instance, in the synthetic example for E2ST model shown in Section 5.1, $\hat{\beta}_l$ can be estimated for β_l since the edge, 2Star and triangle statistics are specified. There are various approaches for parameter estimation.

Maximum likelihood Maximum likelihood is a popular approach for parameter estimation in random graph models. A complication arises because its probability mass function from Eq.(1),

$$q(X = x) = \frac{1}{\kappa_n(\beta)} \exp\left(\sum_{l=1}^{k} \beta_l t_l(x)\right).$$

involves a normalisation constant $\kappa_n(\beta) = \sum_x \exp\{\sum_{l=1}^{k} \beta_l t_l(x)\}$ which is generally intractable and needs to be estimated for performing MLE. For this task, Markov chain Monte-Carlo maximum likelihood estimation (MCMCMLE) for ERGM has been developed, see Snijders [2002]. When the network size is large, accurate estimation for the normalised $\kappa_n(\beta)$ requires large amount of Monte-Carlo samples and is hence computationally expensive.

Maximum pseudo-likelihood estimator To alleviate the problem associated with the normalising constant, Maximum Pseudo-likelihood Estimation (MPLE) [Besag, 1975] has been suggested for ERGMs, see Strauss and Ikeda [1990] and also Schmid and Desmarais [2017]. MPLE factorises the conditional edge probability to approximate the exact likelihood,

$$q(x) = \Pi_{s\in[N]} q(x^s|x_{-s}). \tag{14}$$

For ERGMs the conditional distribution $q(x^s|x_{-s})$ does not involve the normalising constant and can hence be computed more efficiently than the MLE. However, in general the MPLS is not consistent for ERGMs as the edge indicator variables are generally not independent. The consistency of MPLE for Boltzmann machines is shown in Hyvärinen [2006]. A thorough comparison of MCMCMLE and MPLE estimation in ERGMs can be found in Van Duijn et al. [2009].

Contrastive divergence Estimation based on contrastive divergence (CD) [Hinton, 2002] has also been used for ERGM estimation [Hunter and Handcock, 2006]. Contrastive divergence runs a small number of Markov chains simultaneously for T steps and estimates the gradient based on the differences between initial values and values after T steps in order to find a maximum. Convergence results for exponential family models are shown in Jiang et al. [2018]. CD can provide a useful balance between computationally expensive but accurate MCMCMLE and fast but inconsistent MPLE.

B.2 Kernel Stein discrepancies and kernel-based nonparametric hypothesis testing

The task of hypothesis testing involves the comparison of distributions p and q that are significantly different with respect to the size of the test, denoted by α. In nonparametric tests, the distributions are not assumed to be in any parametric families and test statistics are often based on ranking of observations. In contrast, parametric tests, such as a Student t-test or a normality test, assume a pre-defined parametric family to be tested against and usually employ a particular summary statistics such as means or standard deviations. Recent advances in nonparametric test procedures introduce RKHS functions which can be rich enough to distinguish distributions whenever they differ. Below we detail two instances which are relevant for the main paper.

We start with a terse review of **kernel Stein discrepancy (KSD) for continuous distributions** developed to compare and test distributions [Gorham and Mackey, 2015, Ley et al., 2017]. Let q be a smooth probability density on $\mathbb{R}^d$ that vanishes at the boundary. The operator $\mathcal{A}_q : (\mathbb{R}^d \to \mathbb{R}^d) \to (\mathbb{R}^d \to \mathbb{R})$ is called a *Stein operator* if the following *Stein identity* holds: $\mathbb{E}_q[\mathcal{A}_q f] = 0$, where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a bounded smooth function from a set $\mathcal{F}$ which is often called a *Stein class* of functions. Many choices of Stein operators and Stein classes are possible, see for example Mijoule et al. [2021]. A Stein operator $\mathcal{A}_q$ for q a continuous differentiable density on $\mathbb{R}^d$ is defined for differentiable real-valued functions f on $\mathbb{R}^d$ by

$$\mathcal{A}_q f(x) = \sum_{i=1}^{d} \left(f_i(x) \frac{\partial}{\partial x^i} \log q(x) + \frac{\partial}{\partial x^i} f_i(x) \right), \tag{15}$$

see [Chwialkowski et al., 2016, Liu et al., 2016]. This Stein operator is also called Langevin-diffusion Stein operator [Barp et al., 2019]. That the Stein operator $\mathcal{A}_q$ only requires the derivatives of $\log q$ and thus does not involve computing the normalisation constant of q renders it particularly useful for dealing with unnormalised models [Hyvärinen, 2005]. For our purposes it is convenient to take as Stein class $\mathcal{F} = B_1(\mathcal{H})$, the unit ball of a large enough RKHS with bounded kernel K. The kernel Stein discrepancy (KSD) between two densities p and q based on $\mathcal{A}_q$ is defined as

$$\mathrm{KSD}(p\|q, \mathcal{H}) = \sup_{f \in B_1(\mathcal{H})} \mathbb{E}_p[\mathcal{A}_q f]. \tag{16}$$

Under mild regularity conditions, for the Langevin-diffusion operator, $\mathrm{KSD}(p\|q, \mathcal{H}) \geq 0$ and $\mathrm{KSD}(p\|q, \mathcal{H}) = 0$ if and only if $p = q$ [Chwialkowski et al., 2016], in which case KSD is a proper discrepancy measure between probability densities.

The KSD in Eq.(16) can be used to test the model goodness-of-fit as follows. One can show that $\mathrm{KSD}^2(p\|q, \mathcal{H}) = \mathbb{E}_{x,\tilde{x}\sim p}[h_q(x, \tilde{x})]$, where x and $\tilde{x}$ are independent random variables with density p and $h_q(x, \tilde{x})$ is given in explicit form which does not involve p,

$$h_q(x, \tilde{x}) \quad = \quad \langle \mathcal{A}_q K(x, \cdot), \mathcal{A}_q K(\cdot, \tilde{x}) \rangle_{\mathcal{H}} . \tag{17}$$

Given a set of samples $\{x_1, \ldots, x_n\}$ from an unknown density p on $\mathbb{R}^d$, to test whether $p = q$ the statistic $\mathrm{KSD}^2(p\|q, \mathcal{H})$ can be empirically estimated by independent samples from p using a U- or V-statistic. The critical value is determined by bootstrap based on weighted chi-square approximations for U- or V-statistics. For goodness-of-fit tests of discrete distributions when i.i.d. samples are available, a kernel discrete Stein discrepancy (KDSD) has been proposed in Yang et al. [2018].

Goodness-of-fit Testing assesses the null hypothesis $\mathfrak{H}_0 : p = q$ against the general alternative $\mathfrak{H}_1 : p \neq q$ when the target distribution q is explicitly specified. Given sample(s) from the *unknown* distribution p and an explicit density q, $\mathfrak{H}_0$ is assessed using a chosen test statistic, usually a discrepancy measure, $D(q\|p)$, between p and q, which can be estimated empirically. Kernel-based hypothesis tests on goodness-of-fit for continuous distributions q use the kernel Stein discrepancy (KSD) in Section B.2 as the test statistic. Given samples $x_1, \ldots, x_n$ from the *unknown* density p, $\mathrm{KSD}^2(p\|q, \mathcal{H})$ in Eq.(16) is estimated via the V-statistic

$$\widehat{\mathrm{KSD}}^2(p\|q, \mathcal{H}) = \frac{1}{n^2} \sum_{i,j} h_q(x_i, x_j)$$

with $h_q(x_i, x_j) = \langle \mathcal{A}_q K(x_i, \cdot), \mathcal{A}_q K(x_j, \cdot) \rangle_{\mathcal{H}}$ from Eq.(17). The null distribution of this test statistic involves integral operators that are not available in closed form; often it is simulated using a wild-bootstrap procedure [Chwialkowski et al., 2014]. With the (simulated) null distribution, the critical value of the test can be estimated to decide whether the null hypothesis is rejected at test level α. In this way, a general method for nonparametric testing of goodness-of-fit on $\mathbb{R}^d$ is obtained, which is applicable even for models with an intractable normalising constant.

Two-sample Testing assesses whether two sets of samples are drawn from the same distribution; instead of q being available in density form as in the goodness-of-fit setting, q is only accessible through samples. Maximum mean embedding (MMD) test are often used for this two-sample problem [Gretton et al., 2007]. These tests are based on the kernel mean embedding of a distribution,

$$\mu_p := \mathbb{E}_{x\sim p}[k(x, \cdot)] = \int_{\mathcal{X}} k(x, \cdot) dp(x) \in \mathcal{H}, \tag{18}$$

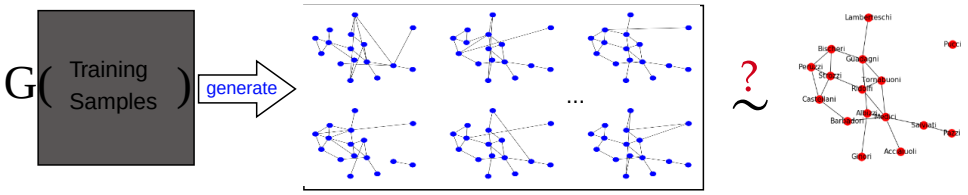


Figure 3: Assessing trained graph generators.

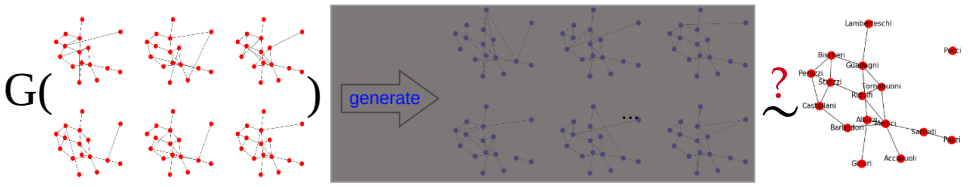


Figure 4: Criticising training quality for generative models.

whenever the function μ_p exist. Similar to KSD, MMD takes the supremum over unit ball RKHS functions;

$$\mathrm{MMD}(p\|q) = \sup_{f \in B_1(\mathcal{H})} \left| \mathbb{E}_p[f] - \mathbb{E}_q[f] \right| = \|\mu_p - \mu_Q\|_{\mathcal{H}}. \tag{19}$$

With samples $x_1, \dots x_m \sim p$ and $y_1, \dots, y_n \sim q$, MMD can be estimated empirically via U-statistics,

$$\widehat{\mathrm{MMD}}^2_u(p\|q) = \frac{1}{m(m-1)} \sum_{i \neq i'} k(x_i, x_{i'}) + \frac{1}{n(n-1)} \sum_{j \neq j'} k(y_j, y_{j'}) - \frac{2}{mn} \sum_{ij} k(x_i, y_j). \tag{20}$$

In such kernel-based two-sample tests, the null distribution can be obtained via a permutation procedure [Gretton et al., 2007]; this procedure can be more robust compared to a wild-bootstrap procedure, especially when the kernels need to be optimised [Gretton et al., 2012, Jitkrittum et al., 2016, Liu et al., 2020, 2021].

The two-sample procedure can also be applied to verify model assumptions when the model is not directly accessible through its distribution but through generated samples. Such a strategy has been considered as benchmark testing procedure in various studies for goodness-of-fit tests [Jitkrittum et al., 2017, Xu and Matsuda, 2020, 2021]. Despite lower test power compared to the corresponding state-of-the-art KSD-based tests and higher computational cost due to additional empirical estimation for the distribution q, these MMD-based tests are competitive with a simpler derivation in complicated testing scenarios [Xu and Matsuda, 2020, 2021], and they can outperform non-kernel based goodness-of-fit tests as discussed in Xu and Matsuda [2020].

C Visual illustrations of the assessment procedures

While AgraSSt is illustrated in Figure.1, we provide an additional visualisation emphasising different tasks for which AgraSSt can be applied. In Section 4.1 in the main text, we mentioned two features of our proposed AgraSSt procedure:

1. Regardless of the learning or training procedures (masked in grey in in Figure.3), AgraSSt can test a given generator G that is only accessible through its generated samples as shown in Figure.3. In this setting, we do not need to know how the generator G is obtained; the focus is the assessment of a particular generator G itself.
2. Moreover, we are also interested in understanding the quality and capability of training procedures of (deep) generative models. As illustrated in Figure.4, a generator G is trained on samples from the same distribution as the input graph, e.g. ERGMs. The focus in this setting is to assess the training procedure of the generative model. (The samples generated are masked in grey.) For instance, for G trained from the Florentine marriage network

[Padgett and Ansell, 1993], we may like to assess whether the generative model can be trained to generate graphs that resemble the Florentine marriage network.

D Additional experimental results and discussions

D.1 Generating reliable samples

To illustrate how AgraSSt can be used to select sample batches, Figure.5 shows three sample batches of size 8 for the Karate club network of Zachary [1977], including the corresponding p-values for the displayed sample batches. Here we would expect to detect some community structure in the networks; visually, only the sample batch from CELL captures this feature at least to some extent and has p-value which would not lead to rejection at the 5% level. This finding chimes with the results from Table 2; AgraSSt rejects both GraphRNN and NetGAN as synthetic data generators, but does not reject CELL.

D.2 Additional case study: Padgett's Florentine network

Padgett's Florentine network [Padgett and Ansell, 1993]. has 16 vertices and 20 edges; in Xu and Reinert [2021] the hypothesis that it is an instance of a $G(n, p)$ model could not be rejected.

	AgraSSt	Deg	MDdeg	TV_deg
GraphRNN	0.01	0.11	0.26	0.03
NetGAN	0.16	0.18	0.09	0.06
CELL	0.23	0.36	0.69	0.18

Table 3: p-values for models trained from Florentine marriage network; 100 samples to simulate the null; rejected null at significant level $\alpha = 0.05$ is marked red.

The p-values for different tests are shown in Table.3. The Florentine marriage network has edge density $q = 0.167$, while the trained CELL has $\widehat{q} = 0.165$ which is a close approximation. GraphRNN generates graphs with higher edge density $\widehat{q} = 0.188$. NetGAN generate samples with $\widehat{q} = 0.176$, not too different from the null, which is not rejected at $\alpha = 0.05$.

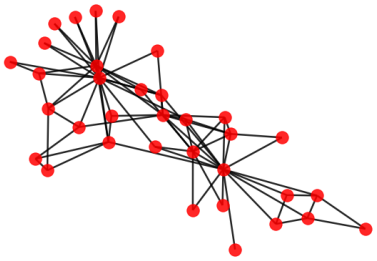
Sample batch selection With CELL being deemed a good generator for the Florentine marriage network, we generate a sample batch of size 30 and check the sample quality. Most sample batches produce a p-value above $\alpha = 0.05$ until the 8th batch, which has p-value $0.03 < \alpha$. AgraSSt would recommend not taking this batch. A visual illustration is shown in Figure.6.

To investigate these batches, we note that the Florentine marriage network has 3 triangles, while the batch being rejected has a significantly lower average number of triangles, namely 1.2. Despite a well estimated edge density, this batch produces a low p-value. This batch, identified by AgraSSt as less reliable, may not be very suitable for downstream tasks and it may be better to generate another batch instead. In contrast, the batch with p-value 0.75 has 2.28 triangles on average while the batch with p-value 0.37 has 2.04 triangles on average; these averages are closer to the observed number of triangles.

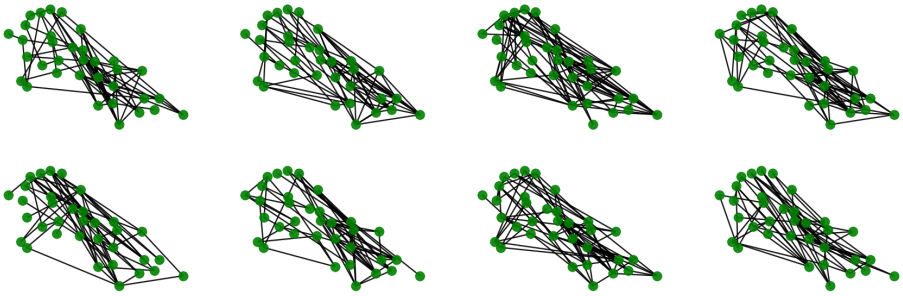
D.3 Experiments with other network statistics

AgraSSt can incorporate any user-defined network statistics. Table 4 and Table 5 show additional results in the settings of Figure 2(b) and Table 1, respectively. As AgraSSt network statistics $t(x_{-s})$, we introduce D3, which for $s = (i, j)$ is the multivariate statistics $t(x_{-s}) = (edges((i, j)), deg(i), deg(j))$, and we introduce Tri, which is the number of common neighbours of i and j; we use the same notation as in Section 5.1.2, calculating the degree and the edges with (i, j) removed. The edge based AgraSSt from the main text is added in grey for comparison.

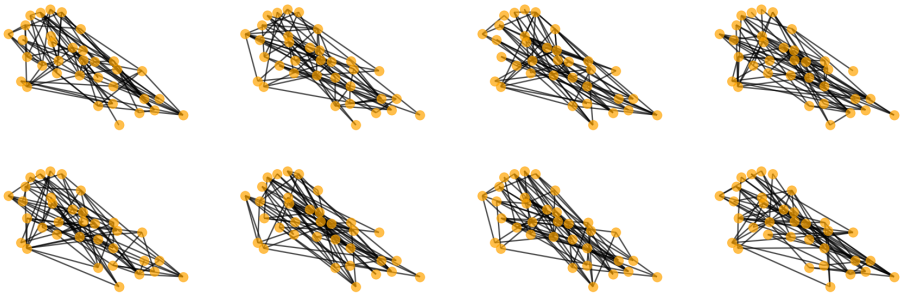
In the Florentine network example, D3 has p-values 0.04 for GraphRNN, 0.11 for NetGAN, and 0.74 for CELL. Tri has p-values 0.02 for GraphRNN, 0.01 for NetGAN, and 0.12 for CELL. Overall, the results are mainly comparable to using AgraSSt based on the number of edges, although Tri rejects



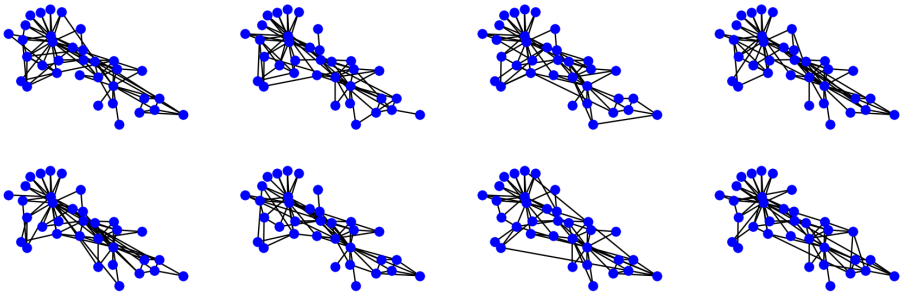
(a) The Karate Club network (vertices in red)



(b) Samples generated from GraphRNN model trained on Karate Club network (vertices in green)



(c) Samples generated from NetGAN model trained on Karate Club network (vertices in orange)



(d) Samples generated from CELL model trained on Karate Club network (vertices in blue) 0.26

Figure 5: The Karate Club network Zachary [1977] and three sample batches of size 8 from different graph generators. The p-value for GraphRNN samples in (b) is 0.00, for NetGAN samples in (c) the p-value is 0.01; for CELL samples in (d) the p-value is 0.26.

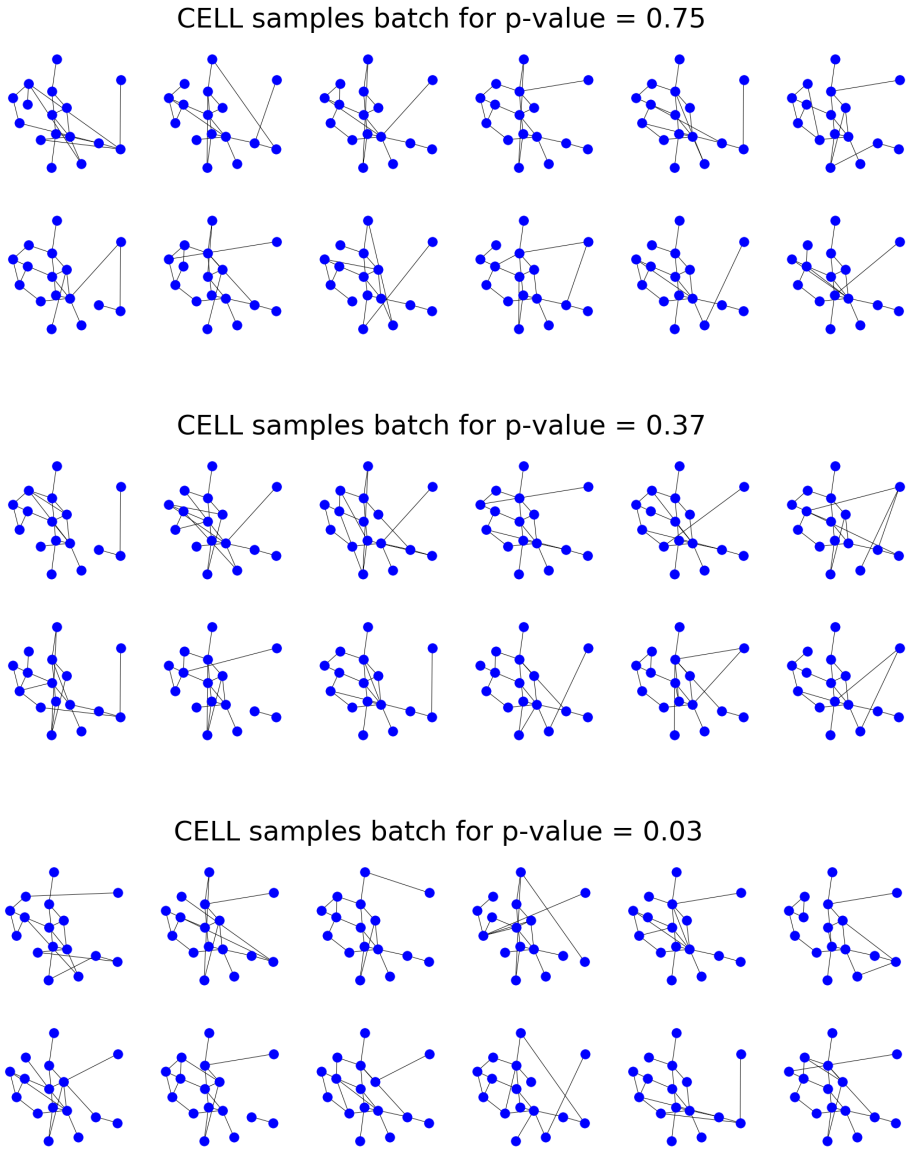


Figure 6: Samples from small size batches generated from CELL trained on the Florentine marriage network, with AgraSSt p-values. The first two batches would be deemed suitable by AgraSSt, while AgraSSt would not accept the third sample at the 5% significance level.

perturbed β_2	-0.60	-0.40	-0.20	0.00	0.20
AgraSSt_D3	0.93	0.87	0.60	0.06	1.00
AgraSSt_Tri	0.82	0.71	0.35	0.07	1.00
AgraSSt (main)	0.95	0.89	0.68	0.04	1.00

Table 4: Rejection Rate for the setting in Figure 2(b).

Models	GraphRNN	NetGAN	CELL	MC
AgraSSt_D3	0.31	0.66	0.10	0.03
AgraSSt_Tri	0.28	0.32	0.12	0.06
AgraSSt (main)	0.42	0.81	0.05	0.04

Table 5: Rejection Rate for the setting in Table 1.

NetGAN for the Florentine marriage network, thus picking up on NetGAN struggling to reproduce local clustering.

D.4 Additional discussions on distance-based test statistics

A classical approach for goodness-of-fit testing in ERGMs is the graphical test by Hunter et al. [2008]. The idea is to simulate sample graphs under the null distribution statistics and create box plots of some relevant network statistics; add to these plots the network statistics in the observed network, as a solid line for comparison, as illustrated in Figure.7. The box plot is used to check whether the observed network is "very different" from the simulated null samples. This graphical test procedure can be translated into Monte Carlo tests. It is natural to adapt such procedure to implicit models from which samples can be obtained. Figure.7 plots standard network statistics from Hunter et al. [2008] for samples from a fitted $G(n, p)$ generator (ER Approximate) and a learned GraphRNN generator of the Florentine marriage network. The bold black line indicates the distribution of statistics for the Florentine marriage network.

The distribution of network statistics is then quantified via Total Variation (TV) distance [Xu and Reinert, 2021], based on which goodness-of-fit testing with p-values can be conducted. We find that while the fitted ER generator shows a reasonable fit for all summary statistics, the GraphRNN generator does not match the Florentine marriage network very well for dyad-wise shared partners and the triad census.

D.5 Efficiency results

Table.6 presents the computational runtime (RT) and the test construction time (CT) for AgraSSt and its comparison methods from Section 5.1.1 with the simulation setup as in Section 5.1.2. As a measure of accuracy, the variance (Var) of the simulated (or estimated) test statistics under the null distribution is also included.

The parameter estimation in **Param** depends on a computationally efficient method which is based on MPLE [Schmid and Desmarais, 2017] in Eq.(14). **AgraSSt** takes longer to compute mainly due to the computation of graph kernels; here we use a Weisfeiler-Lehman kernel [Shervashidze et al., 2011]. We note that for implicit models, the estimation step in AgraSSt relies on generating samples from the model so that the the computational advantage[4] of the Stein based test over graphical goodness-of-fit tests[5] reduces compared to gKSS from Xu and Reinert [2021]. **MDdeg** is computationally expensive due to the estimation of an inverse covariance matrix. While providing fast computation and estimation, **Deg** and **Param** sacrifice test power through a large variance of the test statistics. Estimating the full degree distribution, the total variation distance method **TV_deg**, based only on

[4]These results on gKSS are shown in Supplementary Material D in Xu and Reinert [2021].

[5]The graphical test [Hunter et al., 2008] is computed based on generating a large amount of samples from the null distribution.

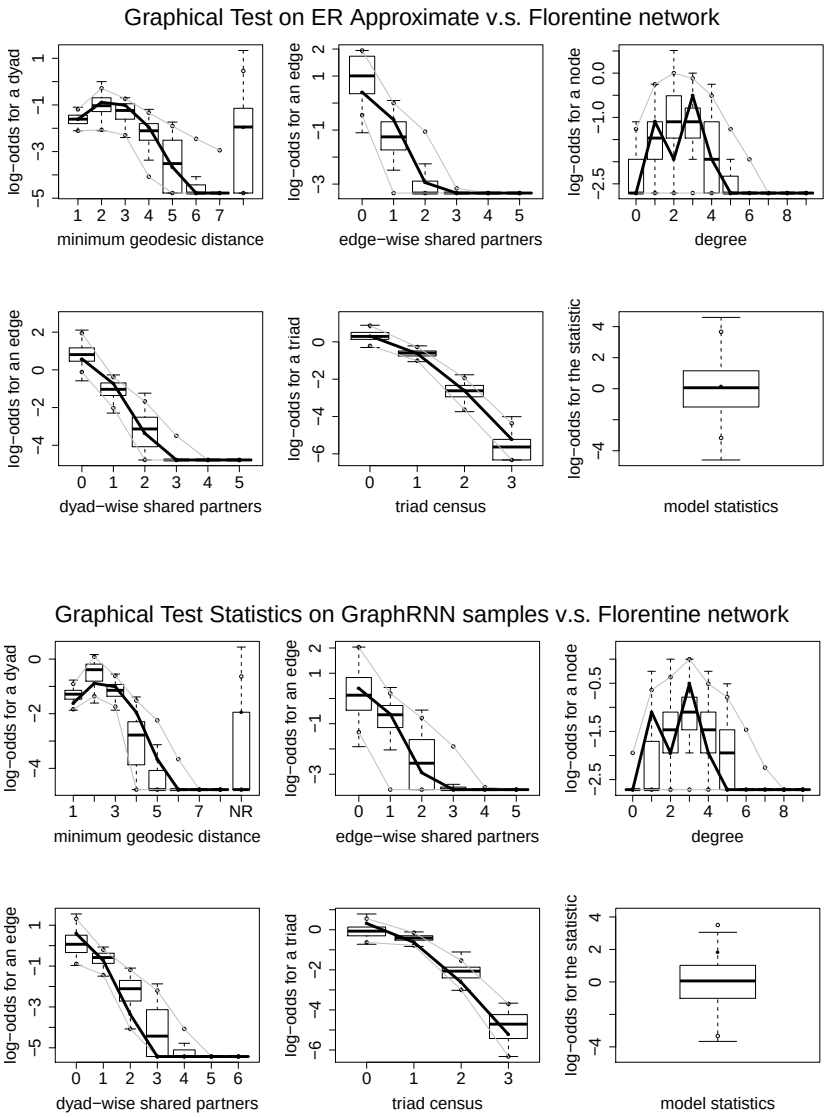


Figure 7: Graphical test illustrations on samples from generators learned on the Florentine marriage network

degrees, is competitive with AgraSSt; we recall that in our simulation results from Section 5.1.2 **TV_deg** was less powerful than AgraSSt. Here **MDdeg** is outperformed by the other test statistics.

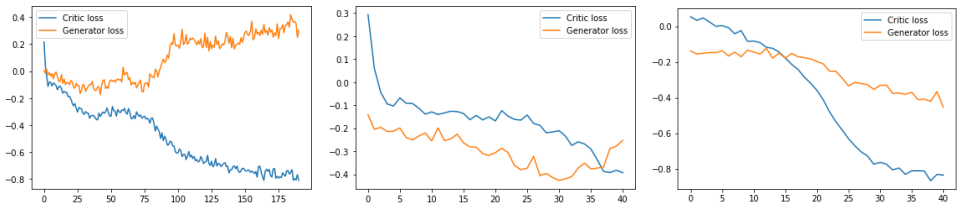
	AgraSSt	Deg	Param	MDdeg	TV_deg
RT(s)	0.141	0.0006	0.014	0.831	0.002
CT(s)	28.656	0.277	2.963	162.912	0.555
Var	0.23	8.38	1.43	15.84	0.28

Table 6: Computational efficiencies and uncertainty in estimates. RT: runtime for one test; CT: construction time for the test class, including generating 500 samples for relevant estimation and 200 samples for simulating from the null distribution; Var: the estimated variance under the simulated null distribution. Both RT and CT are in seconds.

D.6 Additional implementation details

For GraphRNN, we use batch size 128, 1000 epochs for training, 100 for testing, and learning rate 0.003. For CELL, we use learning rate 0.01, and weight decay 10^{-7}. For NetGAN, we use batch size 128, 50 epochs, generator size and discriminator size both 128, and learning rate 0.0003.

We note that training NetGAN [Bojchevski et al., 2018] with the Florentine network and with the Karate Club network may encounter some generator instability and hence early stopping can be useful. Without early stopping, the training loss for the generator increases during training, although it should be decreasing. Figure.8 shows the training loss on the generator in NetGAN as well as on the critic (or discriminator) in NetGAN. Figure.8 plots the loss every 200 training epochs. We can see from Figure.8(a) that the generator loss starts to be unstable and then increases after 50 points, i.e. 10,000 epochs. Hence we use only 10,000 epochs for training.



(a) Florentine network (b) Florentine network with early stopping (c) Karate Club network

Figure 8: Training loss (y-axis) for NetGAN [Bojchevski et al., 2018]; plotted against every 200 training epochs (x-axis).