NetGAN without GAN: From Random Walks to Low-Rank Approximations Supplementary Material

Luca Rendsburg **Holger Heidrich** **Ulrike von Luxburg**

A. Replacing the GAN

In this section, we derive our objective in Eq. (4) by inspecting the high-level structure of NetGAN. The learnable projection matrices are given by $W_{\text{down}} \in \mathbb{R}^{N \times H}$ and $W_{\text{up}} \in \mathbb{R}^{H \times N}$ with $H \ll N$. Given the current node v_t as a one-hot vector and suppressing the next memory state m_{t+1} in notation, the generator f_θ can be written as

$$p_{t+1} = f_\theta\left(m_t, v_t\right) = g_\theta\left(m_t, {v_t}^\top W_{\text{down}}\right) W_{\text{up}}\,, \tag{13}$$

where $g_\theta \colon \mathbb{R}^H \to \mathbb{R}^H$ is the part of f_θ that operates on the low-dimensional space. We collect the row vectors $g_\theta\left(m_t, {v_t}^\top W_{\text{down}}\right)$ in a matrix $\widetilde{W}_{\text{down}}(m_t) \in \mathbb{R}^{N \times H}$ and define the product $W(m_t) \coloneqq \widetilde{W}_{\text{down}}(m_t) W_{\text{up}} \in \mathbb{R}^{N \times N}$ to obtain

$$p_{t+1} = {v_t}^\top \widetilde{W}_{\text{down}}(m_t) W_{\text{up}} = {v_t}^\top W(m_t)\,. \tag{14}$$

Therefore, $W(m_t)$ simply serves as logit transition matrix for the random walks. Because of the factorization that defines $W(m_t)$, its rank is at most H.

To derive how exactly we can replace the GAN with a low-rank approximation, we first simplify the update rule in Eq. (14) by dropping the LSTM and with it the dependency on the memory state m_t; this is justified by the Markov property of unbiased random walks. What remains is a matrix W, whose learnable parameters are intertwined with the low-dimensional part g_θ of the generator:

$$p_{t+1} = {v_t}^\top W = g_\theta\left({v_t}^\top W_{\text{down}}\right) W_{\text{up}}\,. \tag{15}$$

Motivated by the assumption that the identity function $\mathrm{Id} \colon \mathbb{R}^H \to \mathbb{R}^H$ can be represented as g_θ, we drop the structural restriction imposed by g_θ, leaving us with $W = W_{\text{down}} W_{\text{up}}$ and update rule

$$p_{t+1} = {v_t}^\top W_{\text{down}} W_{\text{up}}\,. \tag{16}$$

The new update of node v_t is thereby realized by sampling from the categorical distribution of the corresponding row $\sigma(W_{v_t})$, that is, $v_{t+1} \sim \mathrm{Cat}(\sigma(W_{v_t}))$. In this form, training the GAN is equivalent to learning the random walk transition matrix directly from the parametric family $\mathcal{P} = \{\sigma_{\text{rows}}(W) \in \mathbb{R}^{N \times N} \,:\, W \in \mathbb{R}^{N \times N},\, \mathrm{rank}(W) \leqslant H\}$, where σ_{rows} denotes the function that applies σ to each row of a matrix. We then proceed as described in the main paper by learning the transition matrix from this parametric family directly with the maximum likelihood approach.

B. Information-theoretic representation of objective function F

When considering distributions in this section, we let any matrix with positive entries refer to the uniquely determined distribution that is obtained after normalization. We can reformulate our objective F, defined in Eq. (9), in terms of information-theoretic quantities to determine its minimum irrespective of the rank constraint. To do so, we consider node transitions as a random variable $(\boldsymbol{v}, \boldsymbol{w})$ on $[N] \times [N]$. As derived for Eq. (9) in case of node transitions on the input graph, $(\boldsymbol{v}, \boldsymbol{w})$ is distributed according to the adjacency matrix A, wherefore the corresponding conditional distribution of $\boldsymbol{w}|\boldsymbol{v}$ is given by P. The synthetic transition matrix $\sigma_{\text{rows}}(W)$ represents another conditional distribution for $\boldsymbol{w}|\boldsymbol{v}$. From this

perspective, we can reformulate F as

$$\begin{aligned} F(W) = -\sum_{v,w=1}^{N} A_{v,w} \log \sigma_{\text{rows}}(W)_{v,w} &\propto -\mathbb{E}_{(v,w)\sim A}\left[\log \sigma_{\text{rows}}(W)_{v,w}\right] \\ &= -\mathbb{E}_{(v,w)\sim A}\left[\log A_{v,w}\right] + \mathbb{E}_{(v,w)\sim A}\left[\log\left(\frac{A_{v,w}}{\sigma_{\text{rows}}(W)_{v,w}}\right)\right] \\ &= \mathrm{H}_A(\boldsymbol{w}|\boldsymbol{v}) + \mathrm{KL}\left(A(\boldsymbol{w}|\boldsymbol{v}) \,||\, \sigma_{\text{rows}}(W)(\boldsymbol{w}|\boldsymbol{v})\right) . \end{aligned}$$

The first term on the right-hand side is the conditional entropy of the true underlying node transition distribution A and does not depend on W. The second is the conditional relative entropy between the true node transition distribution A, whose conditional is given by P, and the learned conditional $\sigma_{\text{rows}}(W)$. This shows that F is minimized by any W satisfying $\sigma_{\text{rows}}(W) = P$, which makes the low-rank constraint necessary for generalization in the learning step.

C. Experiments

C.1. Graph statistics

Definition of various graph statistics used in this paper. Part of the table is extracted from Bojchevski et al. (2018).

Table 4. Graph statistics for a graph $G = (V, E)$ with $N = |V|$ nodes and $m = |E|$ edges.

GRAPH STATISTIC	COMPUTATION	DESCRIPTION
ASSORTATIVITY	$\frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$	PEARSON CORRELATION OF DEGREES OF CONNECTED NODES, WHERE THE (x_i, y_i) PAIRS ARE THE DEGREES OF CONNECTED NODES.
POWER LAW EXPONENT	$1 + n\left(\sum_{v\in V} \log \frac{d(v)}{d_{\text{min}}}\right)^{-1}$	EXPONENT OF THE POWER LAW DISTRIBUTION, WHERE d_{MIN} DENOTES THE MINIMUM DEGREE IN A NETWORK.
RELATIVE EDGE DISTRIBUTION ENTROPY	$-\frac{1}{\log N}\sum_{v\in V} \frac{d(v)}{2m} \log \frac{d(v)}{2m}$	NORMALIZED ENTROPY OF THE DEGREE DISTRIBUTION, 1 MEANS UNIFORM, 0 MEANS A SINGLE NODE IS CONNECTED TO ALL OTHERS.
GINI COEFFICIENT	$\frac{2\sum_{i=1}^{N} i\hat{d}_i}{N\sum_{i=1}^{N}\hat{d}_i} - \frac{N+1}{N}$	COMMON MEASURE FOR INEQUALITY IN A DISTRIBUTION, WHERE $\hat{d}$ IS THE SORTED LIST OF DEGREES IN THE GRAPH.
CHARACTERISTIC PATH LENGTH	$\frac{1}{N(N-1)}\sum_{u\neq v} d(u,v)$	AVERAGE SHORTEST PATH LENGTH, WHERE $d(u,v)$ IS THE SHORTEST PATH LENGTH BETWEEN NODES u AND v.
SPECTRAL GAP	$\lambda_1(L)$	SMALLEST NON-ZERO EIGENVALUE λ_1 OF THE GRAPH LAPLACIAN $L = D - A$.
MOTIF COUNT	–	NUMBER OF COPIES OF H CONTAINED IN G AS A SUBGRAPH. CONSIDERED MOTIFS ARE WEDGES, TRIANGLES, AND SQUARES.

C.2. Baselines

- **Configuration model.** We randomly sample a fraction of the edges in the input graph (fraction stated in brackets), and then rewire the remaining edges by severing them and randomly matching the stubs. This yields a graph with the same degree distribution as in the input graph. Because the resulting graph is not simple in general, we then remove all loops and multiple edges (with high probability, there are only few of them).
- **Low-rank approximations with respect to Frobenius norm.** A class of graph generative models similar in spirit to our method is the Spectral Graph Forge framework, which is based on performing low-rank approximations of matrices derived from the input adjacency matrix A. The pipeline consists of the following steps:

1. Transform A into any derived matrix $M = M(A)$.
2. Perform a low-rank approximation of M to obtain $\tilde{M}$.
3. Back-transform $\tilde{M}$ to $\tilde{A}$ by applying the inverse of the transformation.
4. Obtain edge-independent model $A^\dagger$ by making $\tilde{A}$ symmetric and then normalizing it.
5. Sample the new adjacency matrix $A' \sim A^\dagger$.

We apply this framework to the adjacency matrix A (no transformation), the random walk transition matrix $P = D^{-1}A$, the symmetric normalized Laplacian $L^{\text{sym}} = I - D^{-\frac{1}{2}}A^{-\frac{1}{2}}$, and the modularity matrix $B = A - dd^\top/(2e(G))$. The rank of the approximation is chosen so that the desired edge overlap with the input graph is reached: on CORA-ML, we use rank 1600 for $M \in \{A, P, B\}$ and rank 2520 for $M = L^{\text{sym}}$. For the transition matrix, we choose the same back-transformation $\tilde{A} = \text{diag}(\tilde{\pi})\tilde{P}$ as in our method instead of $\tilde{A} = D\tilde{P}$, which uses the degree matrix of the input graph. Given $\tilde{A}$, we proceed like NetGAN and our method with $S = \tilde{A}$. For link prediction, we also use the score matrix.

- **Low-rank approximation with respect to cross-entropy loss.** This baseline is a version of our method CELL, but without the logit space. That is, we solve the optimization problem

$$\begin{aligned} \min_{\tilde{P} \in \mathbb{R}^{N\times N}} & -\sum_{k,l=1}^{N} A_{k,l} \log \tilde{P}_{k,l}\,, \\ \text{s.t.} \quad & \text{rank}(\tilde{P}) \leqslant H \quad \text{and} \quad \tilde{P} \in \mathcal{P}\,, \end{aligned} \tag{17}$$

where $\mathcal{P}$ is the set of stochastic matrices on $\mathbb{R}^{N\times N}$. Similar to how we proceed in CELL, we enforce this constraint with the parametrization

$$\tilde{P} = D(e^C e^D)^{-1} e^C e^D\,, \tag{18}$$

where $C \in \mathbb{R}^{N\times H}$, $D \in \mathbb{R}^{H\times N}$, the exponential function is taken element-wise, and $D(e^C e^D)$ denotes the diagonal matrix of row sums for $e^C e^D$. We then optimize the objective in Eq. (17) over C and D with Adam.

C.3. Stopping criteria

In addition to the rank constraint, CELL and NetGAN both use an early stopping criterion for learning the random walk distribution.

EO-criterion. The EO-criterion is the stopping criterion used in this paper and by NetGAN, and generates graphs with a predefined edge overlap with the input graph (e. g. 50%). To employ it, training is stopped during regular intervals, the edge-independent model is constructed, and a single graph G' is generated. If the fraction of edges $e(G \cap G')/e(G)$ is smaller than the predefined threshold, training is continued, otherwise it is stopped.

VAL-criterion. The VAL-criterion is a stopping criterion proposed by NetGAN and represents an alternative to the edge overlap (EO) criterion used in this paper. It is employed by evaluating the link prediction performance on the validation set during training of the optimization problem, and then stopping the training as soon as the link prediction performance does not improve for a predefined amount of training iterations. For evaluation after training, the test set is used instead of the validation set.

C.4. Example of a bias of NetGAN and CELL: ε-neighborhood graphs

This section demonstrates the concepts discussed in Section 5.2 on ε-neighborhood graphs. These graphs arise by choosing the nodes as points in a metric space, and connecting those pairs of points by an unweighted, undirected edge whose distance is smaller than a constant $\varepsilon > 0$ (see Figure 3 for an illustration). Given an ε-graph as input, there is one major property that a graph generative model should keep intact: edges should occur between points with a small distance in the underlying space, but not for points with a large distance. Figure 3 shows that NetGAN and CELL do not comply with this desired tendency: both algorithms generate long edges. We can easily counteract this mismatch in inductive bias for CELL by extending our loss function with an additional term that penalizes long edges. Because the underlying distances of the metric space are not directly represented in the ε-neighborhood graph anymore, we use its shortest path distances $\mathcal{D} \in \mathbb{R}^{N\times N}$ as a proxy (which is sensible as one can prove that the shortest path distances in ε-graphs converge to the underlying metric

distances (Tenenbaum et al., 2000; Orlitsky, 2005)). We refer to the resulting method as "Local CELL", whose loss function is given by

$$F(W) = -\sum_{k,l=1}^{N} A_{k,l} \log \sigma_{\text{rows}}(W)_{k,l} - \sum_{k,l=1}^{N} A_{k,l} \left[\mathcal{D}_{k,l} \leqslant k\right] \log \sigma_{\text{rows}}(W)_{k,l} \,, \tag{19}$$

where $[\mathcal{A}] = 1$, if statement $\mathcal{A}$ is true, and 0 otherwise. A comparison of NetGAN, CELL, and Local CELL is given in Table 5, see also Figure 3 for an illustration. As expected, NetGAN and CELL generate graphs with long edges, resulting in a large average edge length and small characteristic path length. Local CELL on the other hand is significantly closer to the input graph in this regard without a loss in performance for other statistics. It even improves on some other statistics, because the objective function is more appropriate for this type of graph.

Table 5. Statistics of ε-neighborhood graph and generated graphs from three generative models, averaged over five trials. For the generated graphs, "avg. edge len." is computed only from generated edges that are not present in the input graph.

GRAPH		CHARAC. PATH LEN.	AVG. EDGE LEN.	SPECTRAL GAP	ASSORT-ATIVITY	POWER LAW EXP.	TRIANGLE COUNT	WEDGE COUNT
ε-NEIGHBORHOOD GRAPH		8.97	0.10	2.04e-3	0.75	1.51	2,319	11,557
NETGAN	(51% EO)	3.54	0.41	4.49e-2	0.09	1.50	895	10,638
CELL	(52% EO)	4.01	0.50	3.57e-2	0.38	1.50	1,144	11,030
LOCAL CELL	(52% EO)	5.92	0.21	5.13e-3	0.47	1.51	1,088	11,377

C.5. Additional baseline experiments

Graph statistics and link prediction performance on all data sets described in Section 6 for generated graphs from NetGAN, our method CELL, and baselines, averaged over five trials. Statistics that are matched by model design for the configuration model are indicated as $*$, and cases that are not applicable as $-$. For a visualization and interpretation of the results, see Section C.7.

Table 6. CORA-ML (2,810 nodes, 7,981 edges).

GRAPH		MAX. DEGREE	ASSORT-ATIVITY	TRIANGLE COUNT	SQUARE COUNT	POWER LAW EXP.	CLUSTER-ING COEFF.	CHARAC. PATH LEN.	ROC-AUC SCORE	TIME (IN S)
CORA-ML		238	-0.076	2,802	14,268	1.86	8.26e-2	5.63	1	–
CONF. MODEL	(52% EO)	$*$	-0.053	623	3111	$*$	1.96e-2	4.43	–	1
LR-ADJ	(53% EO)	121	-0.042	444	1,128	1.72	2.78e-2	5.17	0.561	32
LR-TRANS	(57% EO)	139	-0.058	558	1,617	1.77	2.94e-2	5.07	0.709	33
LR-LAP	(52% EO)	167	-0.084	691	1942	1.79	2.79e-2	4.76	0.800	38
LR-MOD	(53% EO)	122	-0.043	437	1,135	1.72	2.75e-2	5.17	0.557	48
LR-CE	(52% EO)	193	-0.068	1,388	6,284	1.79	5.68e-2	5.37	0.950	73
NETGAN	(54% EO)	219	-0.071	1,461	5,555	1.80	5.23e-2	5.13	0.950	7,478
CELL	(53% EO)	204	-0.070	1,396	6,880	1.82	5.07e-2	5.26	0.938	21

Table 7. CITESEER (2,110 nodes, 3,668 edges).

GRAPH	MAX. DEGREE	ASSORT-ATIVITY	TRIANGLE COUNT	SQUARE COUNT	POWER LAW EXP.	CLUSTER-ING COEFF.	CHARAC. PATH LEN.	ROC-AUC SCORE	TIME (IN S)
CITESEER	72	-0.015	483	1,866	2.24	8.70e-2	10.68	1	–
CONF. MODEL (56% EO)	∗	-0.014	108	282	∗	1.95e-2	6.33	–	1
LR-ADJ (57% EO)	34	4.75e-2	89	188	2.09	2.62e-2	8.17	0.608	12
LR-TRANS (57% EO)	36	-0.022	119	364	2.15	3.20e-2	8.58	0.825	8
LR-LAP (57% EO)	48	0.019	108	161	2.18	2.45e-2	7.82	0.362	12
LR-MOD (56% EO)	32	5.33e-4	87	162	2.09	2.67e-2	8.17	0.603	108
LR-CE (56% EO)	47	-0.076	138	549	2.13	3.50e-2	8.57	0.903	19
NETGAN (57% EO)	52	-0.074	361	478	2.15	8.50e-2	9.03	0.951	4,654
CELL (56% EO)	44	-0.093	106	318	2.17	2.54e-2	7.36	0.858	10

Table 8. POLBLOGS (1,222 nodes, 16,779 edges).

GRAPH	MAX. DEGREE	ASSORT-ATIVITY	TRIANGLE COUNT	SQUARE COUNT	POWER LAW EXP.	CLUSTER-ING COEFF.	CHARAC. PATH LEN.	ROC-AUC SCORE	TIME (IN S)
POLBLOGS	298	-0.222	60,873	2,631,731	1.44	0.189	2.82	1	–
CONF. MODEL (52% EO)	∗	-0.140	31,364	1,263,826	∗	0.118	2.72	–	1
LR-ADJ (52% EO)	171	-0.022	15,497	430,846	1.36	0.082	2.66	0.63	1
LR-TRANS (52% EO)	200	-0.114	27,428	918,543	1.40	0.114	2.73	0.861	1
LR-LAP (51% EO)	234	-0.214	19,593	511,781	1.36	0.086	2.55	0.745	2
LR-MOD (51% EO)	170	-0.028	15,528	433,669	1.36	0.082	2.66	0.624	16
LR-CE (54% EO)	248	-0.226	34,942	1,303,305	1.40	0.126	2.66	0.943	17
NETGAN (52% EO)	261	-0.244	37,849	1,438,174	1.41	0.132	2.70	0.950	55,276
CELL (51% EO)	268	-0.243	49,366	2,043,407	1.43	0.160	2.78	0.949	15

Table 9. RT-GOP (4,687 nodes, 5,529 edges).

GRAPH	MAX. DEGREE	ASSORT-ATIVITY	TRIANGLE COUNT	SQUARE COUNT	POWER LAW EXP.	CLUSTER-ING COEFF.	CHARAC. PATH LEN.	ROC-AUC SCORE	TIME (IN S)
RT-GOP	270	-0.135	0	2	4.29	0	14.01	1	–
CONF. MODEL (51% EO)	∗	-0.092	56	241	∗	1.89e-3	5.68	–	1
LR-ADJ (52% EO)	239	-0.117	0	87	3.74	0	12.05	0.559	7
LR-TRANS (55% EO)	328	-0.111	0	139	4.53	0	6.18	0.676	19
LR-LAP (52% EO)	162	-0.070	5	1	3.09	5.15e-4	14.61	0.466	164
LR-MOD (51% EO)	192	-0.101	0	34	3.41	0	21.10	0.550	84
LR-CE (51% EO)	233	-0.122	0	29	3.74	0	20.08	0.874	129
NETGAN (52% EO)	221	-0.112	14	15	3.64	6.74e-4	16.33	0.738	14,800
CELL (51% EO)	253	-0.142	0	6	4.11	0	16.90	0.704	23

Table 10. WEB-EDU (3,031 nodes, 6,547 edges).

GRAPH		MAX. DEGREE	ASSORT-ATIVITY	TRIANGLE COUNT	SQUARE COUNT	POWER LAW EXP.	CLUSTER-ING COEFF.	CHARAC. PATH LEN.	ROC-AUC SCORE	TIME (IN S)
WEB-EDU		99	-0.183	4491	35,423	2.11	0.167	4.56	1	–
CONF. MODEL	(52% EO)	∗	-0.109	873	4,913	∗	0.032	4.59	–	1
LR-ADJ	(53% EO)	58	-0.114	1,096	3,932	1.97	0.081	6.66	0.579	18
LR-TRANS	(53% EO)	166	-0.034	2,692	20,424	2.13	0.120	5.13	0.862	12
LR-LAP	(53% EO)	103	-0.098	514	1,637	2.01	0.028	5.25	0.360	30
LR-MOD	(53% EO)	58	-0.121	989	3,292	1.97	0.075	6.44	0.595	97
LR-CE	(52% EO)	74	-0.136	1,194	4,330	1.99	0.080	6.41	0.994	72
NETGAN	(53% EO)	92	-0.174	1,244	3,022	2.02	0.064	5.51	0.992	11,000
CELL	(54% EO)	63	-0.234	1,176	4,710	2.03	0.069	6.67	0.977	16

C.6. Evolution of graph statistics during training

To compare the generated graphs of CELL and NetGAN for different edge overlaps, we fix all hyperparameters and stop training at regular intervals to compute the graph statistics of the generated graphs. Since we fix the ranks for the low-rank constraint, the generated graphs will not converge to 100% edge overlap. But this area of high edge overlap is of little interest anyway, because the shared edges alone force the generated graphs to reproduce many graph statistics. Note that in order to reduce computational complexity, NetGAN uses less random walks to compute statistics during training (for example to evaluate the stopping criteria): instead of sampling many random walks from the generator, it keeps track of the random walks generated in the last 1,000 iterations during training to build the score matrix. For our method CELL there is no such distinction, we complete our pipeline as described in the main paper.

Aside from few exceptions, for example the triangle count and the related clustering coefficient on CITESEER, we observe the same behavior as described in Section 6.2 of the main paper: after a short initialization phase, CELL and NetGAN display comparable behavior.

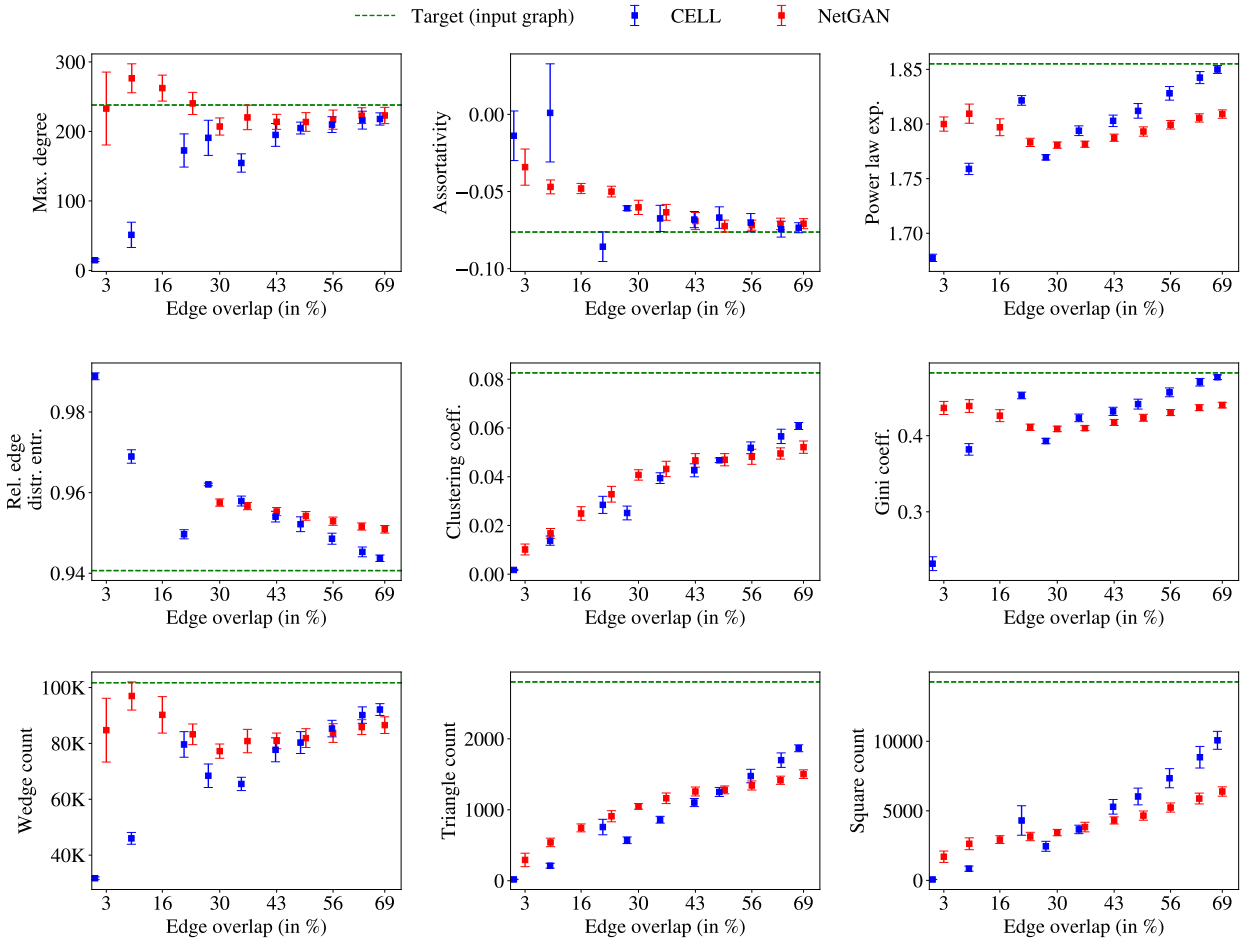

Figure 5. Graph statistics during training for NetGAN and CELL on CORA-ML, plotted against edge overlap.

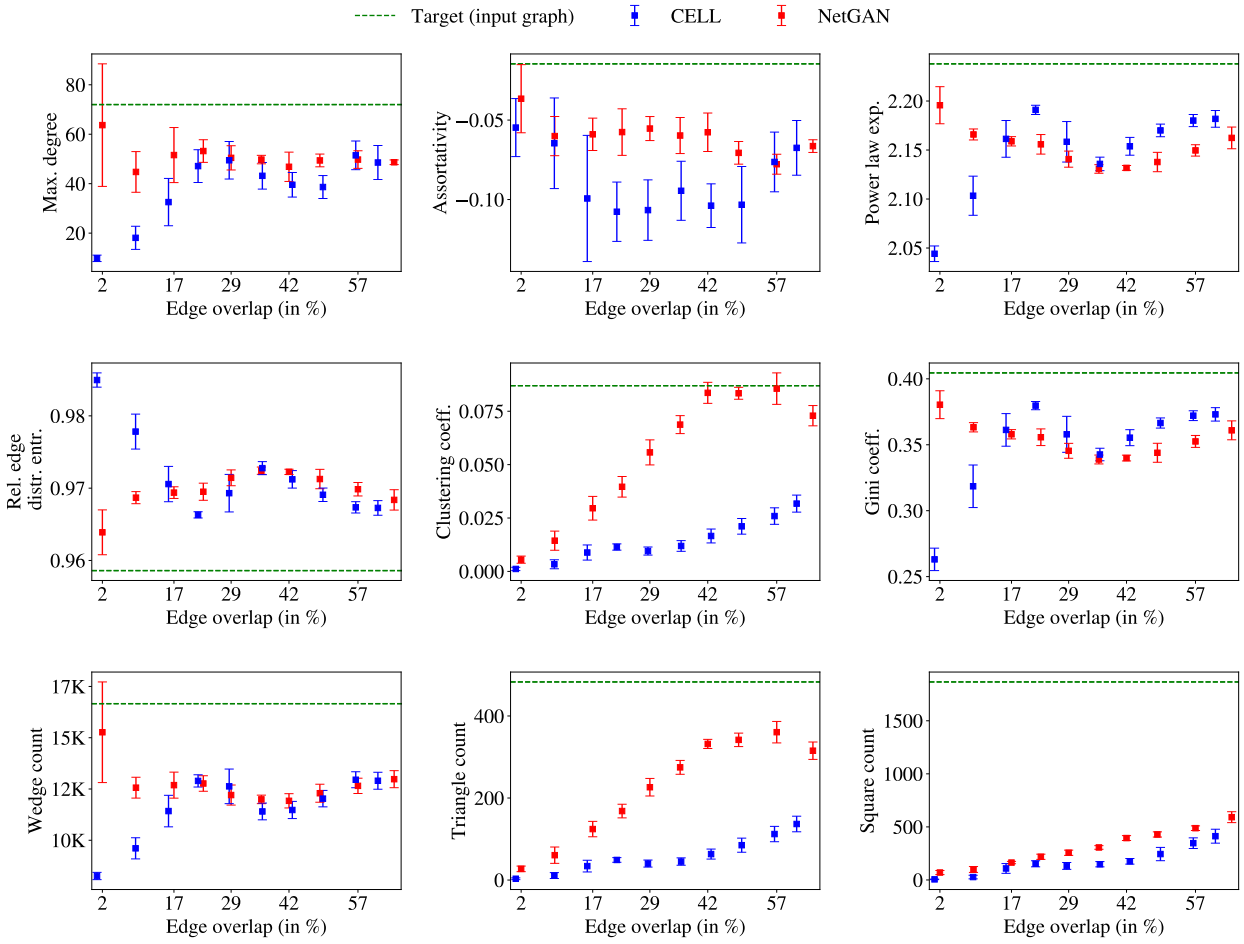

Figure 6. Graph statistics during training for NetGAN and CELL on CITESEER, plotted against edge overlap.

C.7. Comparison of relative errors

For a graph statistic s on the input graph, let $s_{\mathrm{M}} = (s_{\mathrm{M}}^{(1)}, \ldots, s_{\mathrm{M}}^{(K)})$ denote the estimates of model M for s in K trials. The average relative error is then defined by $s_{\mathrm{rel}}(\mathrm{M}) = 1/K \sum_{k=1}^{K} \left| s - s_{\mathrm{M}}^{(k)} \right| / |s|$. In Figure 7, we depict the relative errors for NetGAN, CELL, and baselines on a variety of data sets and graph statistics. Small relative errors indicate good performance in the sense that the generated graphs are close to the input graph. Over all data sets, a general trend can be observed: on most instances, the three models NetGAN, CELL, and LR-CE behave similarly and better as compared to the other baselines. Occasional deviations of this behavior might be attributed to the different optimization procedures and the early stopping. For some networks, for example CITESEER, NetGAN seems to outperform CELL, but for others like POLBLOGS, CELL performs better; this reflects that their different optimization procedures might or might not contribute to the goal of learning the network at hand.

C.8. Hyperparameters

For all our considered models except the configuration model and NetGAN, we choose the rank parameter H such that the generated graphs achieve the predefined edge overlap with the input graph. For example on CORA-ML with 2,810 nodes, we choose $H = 1600$ for LR-Adj, LR-Trans, and LR-Mod, $H = 2520$ for LR-Lap, $H = 950$ for LR-CE, and only $H = 9$ for our method CELL. In general, a higher rank increases the ability of the model to generate graphs with a high edge overlap. For NetGAN, we only consider unbiased random walks ($p = q = 1$) with batch size 128 and length 16. The dimensions H_g and H_d for the low-rank projection for generator and discriminator are both 128. Both generator and discriminator have a single hidden layer with 40 hidden units for the generator and 30 hidden units for the discriminator. The temperature τ is annealed from $\tau = 5$ to $\tau = 0.5$ with a multiplicative decay of $1 - 10^{-5}$ every step.

We optimize the methods LR-CE, NetGAN and CELL using Adam. For LR-CE and CELL, we use a learning rate of 0.1 and weight decay of 10^{-7}, and for NetGAN the learning rate is 0.0003 with L_2-regularization of 10^{-7} for the generator and $5 \cdot 10^{-5}$ for the discriminator. The Wasserstein gradient penalty is set to 10.

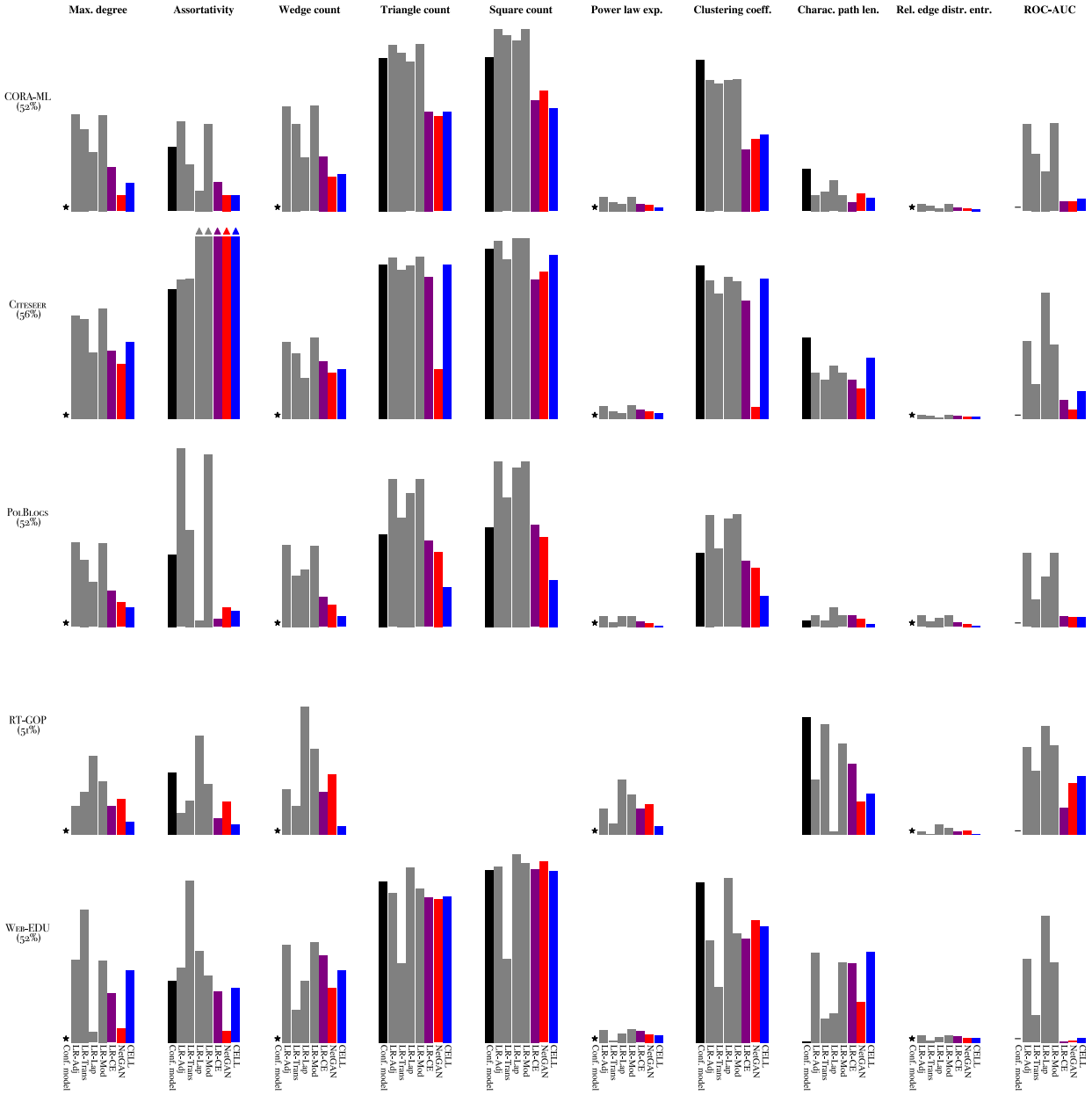

Figure 7. Relative errors of NetGAN, CELL and baselines, trained until the EO-stopping criterion given in brackets, and averaged over five trials. Rows represent the input graphs, columns represent the graph statistics. The y-axis ranges from 0 to 1 in every cell; values that exceed 1 are capped and indicated with an arrow. For the Conf. model, statistics that are matched exactly (0 relative error) are indicated by $\star$, and the non-existent ROC-AUC score is indicated by a $-$. The three statistics triangle count, square count and clustering coefficient for the extremely sparse network RT-GOP are omitted, because their value is zero and the relative errors are not defined (triangle count, clustering coefficient), or it is too small to be produce a meaningful relative error (square count is 2). For the actual graph statistics of generated and input graphs, see Section C.5.