Keep It Simple: Graph Autoencoders Without Graph Convolutional Networks

Guillaume Salha
Deezer Research and Development
LIX, École Polytechnique

Romain Hennequin
Deezer Research and Development

Michalis Vazirgiannis
LIX, École Polytechnique & AUEB

Abstract

Graph autoencoders (AE) and variational autoencoders (VAE) recently emerged as powerful node embedding methods, with promising performances on challenging tasks such as link prediction and node clustering. Graph AE, VAE and most of their extensions rely on graph convolutional networks (GCN) to learn vector space representations of nodes. In this paper, we propose to replace the GCN encoder by a simple linear model w.r.t. the adjacency matrix of the graph. For the two aforementioned tasks, we empirically show that this approach consistently reaches competitive performances w.r.t. GCN-based models for numerous real-world graphs, including the widely used Cora, Citeseer and Pubmed citation networks that became the *de facto* benchmark datasets for evaluating graph AE and VAE. This result questions the relevance of repeatedly using these three datasets to compare complex graph AE and VAE models. It also emphasizes the effectiveness of simple node encoding schemes for many real-world applications.

1 Introduction

Graphs have become ubiquitous, due to the proliferation of data representing relationships or interactions among entities [13, 37]. Extracting relevant information from these entities, called the *nodes* of the graph, is crucial to effectively tackle numerous machine learning tasks, such as link prediction or node clustering. While traditional approaches mainly focused on hand-engineered features [2, 21], significant improvements were recently achieved by methods aiming at directly *learning* node representations that summarize the graph structure (see [13] for a review). In a nutshell, these *representation learning* methods aim at embedding nodes as vectors in a low-dimensional vector space in which nodes with structural proximity in the graph should be close, e.g. by leveraging random walk strategies [11, 27], matrix factorization [4, 24] or graph neural networks [14, 19].

In particular, *graph autoencoders* (AE) [18, 32, 35] and *variational autoencoders* (VAE) [18] recently emerged as powerful node embedding methods. Based on encoding-decoding schemes, i.e. on the design of low dimensional vector space representations of nodes (*encoding*) from which reconstructing the original graph structure (*decoding*) is possible, graph AE and VAE models have been successfully applied to confront several challenging learning tasks, with competitive results w.r.t. popular baselines such as [11, 27, 31]. These tasks include link prediction [12, 18, 25, 33, 28, 29], node clustering [25, 28, 34], matrix completion for inference and recommendation [1, 8] and molecular graph generation [16, 22, 23, 30]. Existing models usually rely on graph neural networks (GNN) to encode nodes into the embedding ; more precisely, most of them implement *graph convolutional networks* (GCN) encoders [8, 12, 15, 18, 20, 25, 28, 29], a model originally introduced in [19].

Accepted Contribution to the Workshop on Graph Representation Learning.
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

In this paper, we analyse the empirical benefit of including GCN encoders in graph AE and VAE w.r.t. more simple heuristics. After reviewing key concepts in Section 2, we introduce simpler versions of graph AE and VAE in Section 3, replacing GCNs by a straightforward linear model w.r.t. the adjacency matrix of the graph, involving a unique weight matrix. In Section 4, we show that these models are empirically competitive[1] w.r.t. GCN-based graph AE and VAE on link prediction and node clustering tasks, for numerous real-world datasets. We discuss these results, aiming at providing insights on settings where GCN encoders should bring (or not) empirical benefits.

2 Preliminaries on Graph (Variational) Autoencoders

Throughout this paper, we consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. We denote by A the adjacency matrix of $\mathcal{G}$, that is either binary or weighted.

Graph Autoencoders Graph autoencoders (AE) [18, 32, 35] are a family of models aiming at mapping (*encoding*) each node $i \in \mathcal{V}$ to a vector $z_i \in \mathbb{R}^d$, with $d \ll n$, from which reconstructing (*decoding*) the graph should be possible. Intuitively, if, starting from the node embedding, the model is also able to reconstruct an adjacency matrix $\hat{A}$ close to the true one, then the z_i vectors should capture some important characteristics of the graph structure. More precisely, the $n \times d$ matrix Z of all z_i vectors is usually the output of a graph neural network (GNN) [3, 7, 19] processing A. To reconstruct the graph, we stack an inner product decoder to this GNN, as in most models [18]. We have $\hat{A}_{ij} = \sigma(z_i^T z_j)$ for all node pairs (i, j), with $\sigma(\cdot)$ denoting the sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$. Therefore, the larger the inner product $z_i^T z_j$ in the embedding, the more likely nodes i and j are connected in $\mathcal{G}$ according to the AE. To sum up, we have $\hat{A} = \sigma(ZZ^T)$ with $Z = \text{GNN(A)}$. Weights of the GNN are trained by gradient descent [9] to minimize a *reconstruction loss* capturing the similarity of A and $\hat{A}$, usually formulated as a weighted cross entropy loss [18].

Graph Variational Autoencoders [18] also extended the *variational autoencoder* (VAE) framework [17] to graph structures. Authors designed a probabilistic model involving latent variables z_i of length $d \ll n$ for each node $i \in \mathcal{V}$, interpreted as node representations in an embedding space. The inference model, i.e. the *encoding* part of the VAE, is defined as:

$$q(Z|A) = \prod_{i=1}^{n} q(z_i|A) \quad \text{where} \quad q(z_i|A) = \mathcal{N}(z_i|\mu_i, \text{diag}(\sigma_i^2)).$$

Gaussian parameters are learned from two GNNs, i.e. $\mu = \text{GNN}_\mu(A)$, with μ the matrix stacking up mean vectors μ_i ; likewise, $\log \sigma = \text{GNN}_\sigma(A)$. Latent vectors z_i are samples drawn from this distribution. From these vectors, a generative model aims at reconstructing (*decoding*) A, leveraging inner products: $p(A|Z) = \prod_{i=1}^{n} \prod_{j=1}^{n} p(A_{ij}|z_i, z_j)$, where $p(A_{ij} = 1|z_i, z_j) = \sigma(z_i^T z_j)$. During training, GNN weights are tuned by maximizing a tractable variational lower bound (ELBO) of the model's likelihood (see [18] for details) by gradient descent, with a Gaussian prior on the distribution of latent vectors, and using the *reparameterization trick* from [17].

Graph Convolutional Networks While the term *GNN encoder* is generic, a majority of successful applications and extensions of graph AE and VAE [8, 12, 15, 18, 20, 25, 28, 29] actually relied on *graph convolutional networks (GCN)* [19] to encode nodes, including the seminal models from [18]. In a GCN with L layers ($L \geq 2$), with input layer $H^{(0)} = I_n$ and output layer $H^{(L)}$ ($H^{(L)} = Z$ for AE, and μ or $\log \sigma$ for VAE), embedding vectors are iteratively updated, as follows:

$$H^{(l)} = \text{ReLU}(\tilde{A}H^{(l-1)}W^{(l-1)}), \text{ for } l \in \{1, ...L-1\} \quad \text{and} \quad H^{(L)} = \tilde{A}H^{(L-1)}W^{(L-1)},$$

where $\tilde{A} = D^{-1/2}(A + I_n)D^{-1/2}$. D is the diagonal degree matrix of $A + I_n$, and $\tilde{A}$ is therefore its symmetric normalization. At each layer, each node averages representations from its neighbors (that, from layer 2, have aggregated representations from their own neighbors), with a ReLU activation: $\text{ReLU}(x) = \max(x, 0)$. Matrices $W^{(0)}, ..., W^{(L-1)}$, whose dimensions can vary, are weight matrices to tune. GCN became a popular encoding scheme, thanks to its relative simplicity w.r.t. [3, 7] and thanks to the linear complexity w.r.t. m of evaluating each layer [19]. Last, GCN models can also leverage node-level features, summarized in an $n \times f$ matrix X, in addition to the graph structure. In such setting, the input layer becomes $H^{(0)} = X$ instead of the identity matrix I_n.

[1] We publicly release the code of these experiments at: https://github.com/deezer/linear_graph_autoencoders

Table 1: Link prediction on Cora, Citeseer and Pubmed. Performances of linear graph AE/VAE are underlined when reaching competitive results w.r.t. GCN-based models from [18] ($\pm$ 1 st. dev.).

Model	Cora (n = 2 708, m = 5 429)		Citeseer (n = 3 327, m = 4 732)		Pubmed (n = 19 717, m = 44 338)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear AE (ours)	83.19 $\pm$ 1.13	87.57 $\pm$ 0.95	77.06 $\pm$ 1.81	83.05 $\pm$ 1.25	81.85 $\pm$ 0.32	87.54 $\pm$ 0.28
2-layer GCN AE	84.79 $\pm$ 1.10	88.45 $\pm$ 0.82	78.25 $\pm$ 1.69	83.79 $\pm$ 1.24	82.51 $\pm$ 0.64	87.42 $\pm$ 0.38
3-layer GCN AE	84.61 $\pm$ 1.22	87.65 $\pm$ 1.11	78.62 $\pm$ 1.74	82.81 $\pm$ 1.43	83.37 $\pm$ 0.98	87.62 $\pm$ 0.68
Linear VAE (ours)	84.70 $\pm$ 1.24	88.24 $\pm$ 1.02	78.87 $\pm$ 1.34	83.34 $\pm$ 0.99	84.03 $\pm$ 0.28	87.98 $\pm$ 0.25
2-layer GCN VAE	84.19 $\pm$ 1.07	87.68 $\pm$ 0.93	78.08 $\pm$ 1.40	83.31 $\pm$ 1.31	82.63 $\pm$ 0.45	87.45 $\pm$ 0.34
3-layer GCN VAE	84.48 $\pm$ 1.42	87.61 $\pm$ 1.08	79.27 $\pm$ 1.78	83.73 $\pm$ 1.13	84.07 $\pm$ 0.47	88.18 $\pm$ 0.31
Model	**Cora, with features** (n = 2 708, m = 5 429, f = 1 433)		**Citeseer, with features** (n = 3 327, m = 4 732, f = 3 703)		**Pubmed, with features** (n = 19 717, m = 44 338, f = 500)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear AE (ours)	92.05 $\pm$ 0.93	93.32 $\pm$ 0.86	91.50 $\pm$ 1.17	92.99 $\pm$ 0.97	95.88 $\pm$ 0.20	95.89 $\pm$ 0.17
2-layer GCN AE	91.27 $\pm$ 0.78	92.47 $\pm$ 0.71	89.76 $\pm$ 1.39	90.32 $\pm$ 1.62	96.28 $\pm$ 0.36	96.29 $\pm$ 0.25
3-layer GCN AE	89.16 $\pm$ 1.18	90.98 $\pm$ 1.01	87.31 $\pm$ 1.74	89.60 $\pm$ 1.52	94.82 $\pm$ 0.41	95.42 $\pm$ 0.26
Linear VAE (ours)	92.55 $\pm$ 0.97	93.68 $\pm$ 0.68	91.60 $\pm$ 0.90	93.08 $\pm$ 0.77	95.91 $\pm$ 0.13	95.80 $\pm$ 0.17
2-layer GCN VAE	91.64 $\pm$ 0.92	92.66 $\pm$ 0.91	90.72 $\pm$ 1.01	92.05 $\pm$ 0.97	94.66 $\pm$ 0.51	94.84 $\pm$ 0.42
3-layer GCN VAE	90.53 $\pm$ 0.94	91.71 $\pm$ 0.88	88.63 $\pm$ 0.95	90.20 $\pm$ 0.81	92.78 $\pm$ 1.02	93.33 $\pm$ 0.91

3 Simplifying the Encoding Scheme

Linear Graph AE In this section, we propose to replace the GCN encoder by a simple linear model w.r.t. the normalized adjacency matrix of the graph. In the AE framework, we have:

$$Z = \tilde{A}W \quad \text{then} \quad \hat{A} = \sigma(ZZ^T).$$

Embedding vectors are obtained by multiplying the $n \times n$ normalized adjacency matrix $\tilde{A}$ by a single $n \times d$ weight matrix W, tuned by gradient descent in a similar fashion w.r.t. standard AE. This encoder is a straightforward linear mapping. Contrary to standard GCN encoders (as $L \geq 2$), nodes only aggregate information from their one-step neighbors. If data include node-level features X, the encoding step becomes $Z = \tilde{A}XW$, and W is of dimension $f \times d$.

Linear Graph VAE We adopt the same approach to replace the encoder of graph VAE by:

$$\mu = \tilde{A}W_\mu \text{ and } \log\sigma = \tilde{A}W_\sigma \quad \text{then} \quad \forall i \in \mathcal{V}, z_i \sim \mathcal{N}(\mu_i, \text{diag}(\sigma_i^2)),$$

with similar decoder w.r.t. standard graph VAE. We refer to this simpler model as *linear graph VAE*, and optimize the standard ELBO bound [18] w.r.t. weight matrices W_μ and W_σ by gradient descent. If data include node-level features X, we compute $\mu = \tilde{A}XW_\mu$ and $\log\sigma = \tilde{A}XW_\sigma$.

4 Empirical Analysis and Discussion

To compare linear graph AE and VAE to GCN-based models, we first focus on *link prediction*, as in [18] and most subsequent works. We train models on incomplete versions of graphs where 15% of edges were randomly removed. Then, we create validation and test sets from removed edges (resp. from 5% and 10% of edges) and from the same number of randomly sampled pairs of unconnected nodes. We evaluate the model's ability to classify edges from non-edges, using the mean *Area Under the Receiver Operating Characteristic (ROC) Curve* (AUC) and *Average Precision* (AP) scores on test sets, averaged over 100 runs with different random train/validation/test splits.

Table 1 reports results for the Cora, Citeseer and Pubmed citation graphs from [10], with and without node features corresponding to bag-of-words vectors. These three graphs were used in the original experiments of [18] and then in the wide majority of recent works [12, 15, 18, 20, 25, 26, 28, 29, 33, 34], becoming the *de facto* benchmark datasets for evaluating graph AE and VAE. For standard AE and VAE, we managed to reproduce performances from [18] ; for all models, we detail hyperparameters in annex. In Table 1, we show that linear models consistently reach competitive performances w.r.t. with 2 and 3-layer GCN encoders, i.e. they are at least as good ($\pm$ 1 standard deviation). In some settings, linear AE/VAE are even slightly better (e.g. $+1.25$ points in AUC for linear VAE on Pubmed with features). We did not report performances of deeper models, due to

Table 2: Link prediction on alternative real-world datasets. Performances of linear graph AE/VAE are underlined when reaching competitive results w.r.t. GCN-based models from [18] ($\pm$ 1 st. dev.).

Model	WebKD (n = 877, m = 1 608)		WebKD, with features (n = 877, m = 1 608, f = 1 703)		Hamsterster (n = 1 858, m = 12 534)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear AE (ours)	77.20 $\pm$ 2.35	83.55 $\pm$ 1.81	84.15 $\pm$ 1.64	87.01 $\pm$ 1.48	93.07 $\pm$ 0.67	94.20 $\pm$ 0.58
2-layer GCN AE	77.88 $\pm$ 2.57	84.12 $\pm$ 2.18	86.03 $\pm$ 3.97	87.97 $\pm$ 2.76	92.07 $\pm$ 0.63	93.01 $\pm$ 0.69
3-layer GCN AE	78.20 $\pm$ 3.69	83.13 $\pm$ 2.58	81.39 $\pm$ 3.93	85.34 $\pm$ 2.92	91.40 $\pm$ 0.79	92.22 $\pm$ 0.85
Linear VAE (ours)	83.50 $\pm$ 1.98	86.70 $\pm$ 1.53	85.57 $\pm$ 2.18	88.08 $\pm$ 1.76	91.08 $\pm$ 0.70	91.85 $\pm$ 0.64
2-layer GCN VAE	82.31 $\pm$ 2.55	86.15 $\pm$ 2.03	87.87 $\pm$ 2.48	88.97 $\pm$ 2.17	91.62 $\pm$ 0.60	92.43 $\pm$ 0.64
3-layer GCN VAE	82.17 $\pm$ 2.70	85.35 $\pm$ 2.25	89.69 $\pm$ 1.80	89.90 $\pm$ 1.58	91.06 $\pm$ 0.71	91.85 $\pm$ 0.77

Model	DBLP (n = 12 591, m = 49 743)		Cora-larger (n = 23 166, m = 91 500)		Arxiv-HepTh (n = 27 770, m = 352 807)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear AE (ours)	90.11 $\pm$ 0.40	93.15 $\pm$ 0.28	94.64 $\pm$ 0.08	95.96 $\pm$ 0.10	98.34 $\pm$ 0.03	98.46 $\pm$ 0.03
2-layer GCN AE	90.29 $\pm$ 0.39	93.01 $\pm$ 0.33	94.80 $\pm$ 0.08	95.72 $\pm$ 0.05	97.97 $\pm$ 0.09	98.12 $\pm$ 0.09
3-layer GCN AE	89.91 $\pm$ 0.61	92.24 $\pm$ 0.67	94.51 $\pm$ 0.31	95.11 $\pm$ 0.28	94.35 $\pm$ 1.30	94.46 $\pm$ 1.31
Linear VAE (ours)	90.62 $\pm$ 0.30	93.25 $\pm$ 0.22	95.20 $\pm$ 0.16	95.99 $\pm$ 0.12	98.35 $\pm$ 0.05	98.46 $\pm$ 0.05
2-layer GCN VAE	90.40 $\pm$ 0.43	93.09 $\pm$ 0.35	94.60 $\pm$ 0.20	95.74 $\pm$ 0.13	97.75 $\pm$ 0.08	97.91 $\pm$ 0.06
3-layer GCN VAE	89.92 $\pm$ 0.59	92.52 $\pm$ 0.48	94.48 $\pm$ 0.28	95.30 $\pm$ 0.22	94.57 $\pm$ 1.14	94.73 $\pm$ 1.12

Model	Blogs (n = 1 224, m = 19 025)		Proteins (n = 6 327, m = 147 547)		Google (n = 15 763, m = 171 206)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear AE (ours)	91.71 $\pm$ 0.39	92.53 $\pm$ 0.44	94.09 $\pm$ 0.23	96.01 $\pm$ 0.16	96.02 $\pm$ 0.14	97.09 $\pm$ 0.08
2-layer GCN AE	91.57 $\pm$ 0.34	92.51 $\pm$ 0.29	94.55 $\pm$ 0.20	96.39 $\pm$ 0.16	96.66 $\pm$ 0.24	97.45 $\pm$ 0.25
3-layer GCN AE	91.74 $\pm$ 0.37	92.62 $\pm$ 0.31	94.30 $\pm$ 0.19	96.08 $\pm$ 0.15	95.10 $\pm$ 0.27	95.94 $\pm$ 0.20
Linear VAE (ours)	91.34 $\pm$ 0.24	92.10 $\pm$ 0.24	93.99 $\pm$ 0.10	95.94 $\pm$ 0.16	91.11 $\pm$ 0.31	92.91 $\pm$ 0.18
2-layer GCN VAE	91.85 $\pm$ 0.22	92.60 $\pm$ 0.25	94.57 $\pm$ 0.18	96.18 $\pm$ 0.33	96.11 $\pm$ 0.59	96.84 $\pm$ 0.51
3-layer GCN VAE	91.83 $\pm$ 0.48	92.65 $\pm$ 0.35	94.27 $\pm$ 0.25	95.71 $\pm$ 0.28	95.10 $\pm$ 0.54	96.00 $\pm$ 0.44

significant scores deterioration. These results emphasize the effectiveness of the proposed simple encoding scheme, limiting to linear and first-order interactions, on these datasets. In supplementary materials, we report two additional tables, consolidating our results by reaching similar conclusions when replacing inner products by more complex decoders [12, 29], and for a *node clustering* task.

In addition to Cora, Citeseer and Pubmed, Table 2 reports experiments on eight other real-world graphs with various characteristics. Hyperparameters details are reported in annex. Overall, the linear graph AE and VAE are competitive in six cases out of nine : for the WebKD [10] hyperlinks web graph, with and without node features (1703-dim bag of words vectors), the Hamsterster [5] social network, a larger version of Cora [5], and the DBLP [5] and Arxiv-HepTh [6] citation networks. Such results confirm the empirical effectiveness of simple node encoding schemes, that might appear as a suitable alternative to complex encoders for many real-world applications.

Nonetheless, GCN-based models are outperforming on the Blogs [5] graph of hyperlinks between blogs from the 2004 US election (for VAE on link prediction, and for both AE/VAE on node clustering), on the Proteins [5] network of proteins interactions, and on the Google [5] hyperlinks network of web pages within Google's sites. These datasets are relatively *dense* ; among dense graphs, the benefit of GCN encoders also increases with the *size* of the graph. Last, the *nature* of the graph seems crucial. In *citation graphs*, if a reference A in an article B cited by some authors is relevant to their work, authors will likely also cite this reference A (creating a first order link); therefore, in such graphs the impact of high-order interactions is empirically limited (even on the quite dense Arxiv-HepTh graph). As a consequence, we conjecture that larger and denser graphs with intrinsic high-order interactions (e.g. some web graphs) should be better suited that the sparse medium-size Cora, Citeseer and Pubmed citation networks, when comparing and evaluating complex encoders.

5 Conclusion

We highlighted that, in graph AE/VAE, simple first-order linear encoders are as effective as the popular GCNs on numerous real-world graphs. Theses results are consistent with recent efforts, out of the AE/VAE unsupervised frameworks, to simplify GCNs [36]. Arguing that current benchmark datasets might be too simple, we hope that our analysis will initiate further discussions on the training and evaluation of graph AE/VAE that should, in the future, lead towards their improvement.

References

[1] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2018. Graph convolutional matrix completion. *KDD Deep Learning day*.

[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management (CIKM)*. ACM, 891–900.

[5] KONECT Koblenz Network Collection. 2019. Blogs, Cora-larger, DBLP, Hamsterster, Proteins (Reactome) and Google.com datasets. `http://konect.uni-koblenz.de/networks/`.

[6] SNAP Stanford Large Network Dataset Collection. 2019. Arxiv-HepTH datasets. `http://snap.stanford.edu/data/index.html`.

[7] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] Tien Huu Do and et. al. 2019. Matrix Completion with Variational Graph Autoencoders: Application in Hyperlocal Air Quality Inference. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7535–7539.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[10] LINQS Statistical Relational Learning Group. 2019. Cora, Citeseer, Pubmed and WebKD datasets. `https://linqs.soe.ucsc.edu/data`.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[12] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative Generative Modeling of Graphs. In *International Conference on Machine Learning (ICML)*. 2434–2444.

[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1024–1034.

[15] Po-Yao Huang, Robert Frederking, et al. 2019. RWR-GAE: Random Walk Regularization for Graph Auto Encoders. *arXiv preprint arXiv:1908.04003*.

[16] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. *International Conference on Machine Learning (ICML)*.

[17] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*.

[18] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*.

[20] Sébastien Lerique, Jacob Levy Abitbol, and Márton Karsai. 2019. Joint embedding of structure and features via graph convolutional networks. *arXiv preprint arXiv:1905.08636*.

[21] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7, 1019–1031.

[22] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. Constrained Graph Variational Autoencoders for Molecule Design. *Advances in Neural Information Processing Systems (NeurIPS)*.

[23] Tengfei Ma, Jie Chen, and Cao Xiao. 2018. Constrained Generation of Semantically Valid Graphs via Regularizing Variational Autoencoders. *Advances in Neural Information Processing Systems (NeurIPS)*.

[24] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[25] S Pan, R Hu, G Long, J Jiang, L Yao, and C Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[26] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. *arXiv preprint arXiv:1908.02441*.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[28] Guillaume Salha, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. 2019. A Degeneracy Framework for Scalable Graph Autoencoders. *International Joint Conference on Artificial Intelligence (IJCAI)*.

[29] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. 2019. Gravity-inspired graph autoencoders for directed link prediction. *ACM International Conference on Information and Knowledge Management (CIKM)*.

[30] Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks (ICANN)*. 412–422.

[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. International World Wide Web Conferences Steering Committee, 1067–1077.

[32] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[33] Phi Vu Tran. 2018. Multi-Task Graph Autoencoders. *arXiv preprint arXiv:1811.02798*.

[34] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *ACM Conference on Information and Knowledge Management (CIKM)*. ACM, 889–898.

[35] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[36] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*. 6861–6871.

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

Supplementary Material

This supplementary material provides details on our experiments and two complementary tables.

Annex A - Experimental Setting and Hyperparameters Details

For *link prediction* experiments (Tables 1, 2 and 3), we followed the setting of [18] and trained models on incomplete versions of graphs where 15% of edges were randomly removed. We created validation and test sets from respectively 5% and 10% removed edges and from the same number of randomly sampled pairs of unconnected nodes. As [18], we ignored edges directions when initial graphs were directed. The link prediction task is a binary classification task, consisting in discriminating edges from non-edges in the test set. We note that, due to high performances on the Proteins graph [5] (AUC and AP scores above 99.5% for all models), we complicated the learning task, by only including 10% of edges (resp. 25%) to the training (resp. test) set for this graph.

All models were trained for 200 epochs. For Cora, Citeseer and Pubmed, we set identical hyperparameters w.r.t. [18] in order to reproduce their results, i.e. we had $d = 16$, GCNs included 32-dim hidden layers, and we used Adam optimizer with a learning rate of 0.01. We indeed managed to reach similar scores as [18] when training 2-layer GCN-based AE and VAE with, however, larger standard errors. This difference comes from the fact that we report average scores on 100 different and random train/validation/test splits, while they used fixed dataset splits for all runs (randomness in their results comes from initialization). For other datasets, validation set was used for hyperparameters tuning. We adopted a learning rate of 0.1 for Arxiv-HepTh ; of 0.01 for Blogs, Cora-larger, DBLP, Google and Hamsterster and Proteins (AE models) ; of 0.005 for WebKD (except linear AE and VAE where we used 0.001 and 0.01) and Proteins (VAE models). All models learned 16-dim embeddings i.e. $d = 16$, with 32-dim hidden layers and without dropout.

While running time is not the main focus of this paper, we also note that linear AE and VAE models tend to be 10% to 15% faster than their GCN-based counterparts, as the proposed simple encoders include fewer matrix operations: e.g. 6.03 seconds (vs 6.73 seconds) mean running time for linear graph VAE (vs 2-layer GCN-based graph VAE) on the featureless Citeseer dataset, on an NVIDIA GTX 1080 GPU. As the standard inner-product decoder has a $O(dn^2)$ quadratic time complexity [28], we focused on medium-size datasets (with roughly $< 30K$ nodes) in our experiments ; we nonetheless point out the existence of recent works [28] on scalable graph autoencoders.

Annex B - Experiments on Link Prediction with Alternative Decoders

In Table 3, we report complementary *link prediction* experiments, on variants of graph AE and VAE where we replaced the inner product decoder by two more complex decoding schemes from existing literature: the Graphite model from [12] and the gravity-inspired asymmetric decoder from [29].

We draw similar conclusions w.r.t. Tables 1 and 2, consolidating results from Section 4. For the sake of brevity, we only report results for the Cora, Citeseer and Pubmed graphs, where linear models are mostly competitive, and for the Google graph, were GCN-based graph AE and VAE are outperforming. We stress out that scores from Graphite and gravity-inspired models are *not* directly comparable, as the former ignores edges directionalities while the latter processes directed graphs (i.e. the learning task becomes a *directed* link prediction problem).

Annex C - Experiments on Node Clustering

In Table 4, we report *node clustering* experiments on datasets that include node-level ground-truth communities. We trained models on complete graphs, then ran k-means algorithms (with k-means++ initialization) in node embedding spaces. We compared output clusters to ground-truth communities by computing *adjusted mutual information (AMI)* scores, averaged over 100 runs.

Overall, linear AE and VAE models are competitive w.r.t. their GCN-based counterparts from [18] on the Cora, Cora-larger, Citeseer and Pubmed citation graphs, in which nodes are documents clustered in respectively 6, 70, 7 and 3 topic classes. However, 2-layer and 3-layer GCN-based models are significantly outperforming on the *dense* Blogs graph, where political blogs are classified as either left-leaning or right-leaning. This is consistent w.r.t. insights from Section 4 that GCN-based encoding should bring larger empirical benefits on dense graphs.

Table 3: Link prediction with alternative decoding schemes [12, 29]. Performances of linear graph AE/VAE are underlined when reaching competitive results w.r.t. GCN-based models ($\pm$ 1 st. dev.).

Model	Cora (n = 2 708, m = 5 429)		Citeseer (n = 3 327, m = 4 732)		Pubmed (n = 19 717, m = 44 338)		Google (n = 15 763, m = 171 206)	
	AUC (in %)	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**	**AUC (in %)**	**AP (in %)**
Linear Graphite AE (ours)	83.42 $\pm$ 1.76	87.32 $\pm$ 1.53	77.56 $\pm$ 1.41	82.88 $\pm$ 1.15	80.28 $\pm$ 0.86	85.81 $\pm$ 0.67	94.30 $\pm$ 0.22	95.09 $\pm$ 0.16
2-layer Graphite AE	81.20 $\pm$ 2.21	85.11 $\pm$ 1.91	73.80 $\pm$ 2.24	79.32 $\pm$ 1.83	79.98 $\pm$ 0.66	85.33 $\pm$ 0.41	95.54 $\pm$ 0.42	95.99 $\pm$ 0.39
3-layer Graphite AE	79.06 $\pm$ 1.70	81.79 $\pm$ 1.62	72.24 $\pm$ 2.29	76.60 $\pm$ 1.95	79.96 $\pm$ 1.40	84.88 $\pm$ 0.89	93.99 $\pm$ 0.54	94.74 $\pm$ 0.49
Linear Graphite VAE (ours)	83.68 $\pm$ 1.42	87.57 $\pm$ 1.16	78.90 $\pm$ 1.08	83.51 $\pm$ 0.89	79.59 $\pm$ 0.33	86.17 $\pm$ 0.31	92.71 $\pm$ 0.38	94.41 $\pm$ 0.25
2-layer Graphite VAE	84.89 $\pm$ 1.48	88.10 $\pm$ 1.22	77.92 $\pm$ 1.57	82.56 $\pm$ 1.31	82.74 $\pm$ 0.30	87.19 $\pm$ 0.36	96.49 $\pm$ 0.22	96.91 $\pm$ 0.17
3-layer Graphite VAE	85.33 $\pm$ 1.19	87.98 $\pm$ 1.09	77.46 $\pm$ 2.34	81.95 $\pm$ 1.71	84.56 $\pm$ 0.42	88.01 $\pm$ 0.39	96.32 $\pm$ 0.24	96.62 $\pm$ 0.20
Linear Gravity AE (ours)	90.71 $\pm$ 0.95	92.95 $\pm$ 0.88	80.52 $\pm$ 1.37	86.29 $\pm$ 1.03	76.78 $\pm$ 0.38	84.50 $\pm$ 0.32	97.46 $\pm$ 0.07	98.30 $\pm$ 0.04
2-layer Gravity AE	87.79 $\pm$ 1.07	90.78 $\pm$ 0.82	78.36 $\pm$ 1.55	84.75 $\pm$ 1.10	75.84 $\pm$ 0.42	83.03 $\pm$ 0.22	97.77 $\pm$ 0.10	98.43 $\pm$ 0.10
3-layer Gravity AE	87.76 $\pm$ 1.32	90.15 $\pm$ 1.45	78.32 $\pm$ 1.92	84.88 $\pm$ 1.36	74.61 $\pm$ 0.30	81.68 $\pm$ 0.26	97.58 $\pm$ 0.12	98.28 $\pm$ 0.11
Linear Gravity VAE (ours)	91.29 $\pm$ 0.70	93.01 $\pm$ 0.57	86.65 $\pm$ 0.95	89.49 $\pm$ 0.69	79.68 $\pm$ 0.36	85.00 $\pm$ 0.21	97.32 $\pm$ 0.06	98.26 $\pm$ 0.05
2-layer Gravity VAE	91.92 $\pm$ 0.75	92.46 $\pm$ 0.64	87.67 $\pm$ 1.07	89.79 $\pm$ 1.01	77.30 $\pm$ 0.81	82.64 $\pm$ 0.27	97.84 $\pm$ 0.25	98.18 $\pm$ 0.14
3-layer Gravity VAE	90.80 $\pm$ 1.28	92.01 $\pm$ 1.19	85.28 $\pm$ 1.33	87.54 $\pm$ 1.21	76.52 $\pm$ 0.61	80.73 $\pm$ 0.63	97.32 $\pm$ 0.23	97.81 $\pm$ 0.20

Table 4: Node Clustering on graphs with communities. Performances of linear graph AE/VAE are underlined when reaching competitive results w.r.t. GCN-based models [18] ($\pm$ 1 st. dev.).

Model	Cora (n = 2.708, m = 5 429)	Citeseer (n = 3 327, m = 4 732)	Pubmed (n = 19 717, m = 44 338)	Cora-larger (n = 23 166, m = 91 500)
	AMI (in %)	**AMI (in %)**	**AMI (in %)**	**AMI (in %)**
Linear AE (ours)	26.31 $\pm$ 2.85	8.56 $\pm$ 1.28	10.76 $\pm$ 3.70	40.34 $\pm$ 0.51
2-layer GCN AE	30.88 $\pm$ 2.56	9.46 $\pm$ 1.06	16.41 $\pm$ 3.15	39.75 $\pm$ 0.79
3-layer GCN AE	33.06 $\pm$ 3.10	10.69 $\pm$ 1.98	23.11 $\pm$ 2.58	35.67 $\pm$ 1.76
Linear VAE (ours)	34.35 $\pm$ 1.42	12.67 $\pm$ 1.27	25.14 $\pm$ 2.83	43.32 $\pm$ 0.52
2-layer GCN VAE	26.66 $\pm$ 3.94	9.85 $\pm$ 1.24	20.52 $\pm$ 2.97	38.34 $\pm$ 0.64
3-layer GCN VAE	28.43 $\pm$ 2.83	10.64 $\pm$ 1.47	21.32 $\pm$ 3.70	37.30 $\pm$ 1.07
Model	**Cora, with features** (n = 2 708, m = 5 429, f = 1 433)	**Citeseer, with features** (n = 3 327, m = 4 732, f = 3 703)	**Pubmed, with features** (n = 19 717, m = 44 338, f = 500)	**Blogs** (n = 1 224, m = 19 025)
	AMI (in %)	**AMI (in %)**	**AMI (in %)**	**AMI (in %)**
Linear AE (ours)	47.02 $\pm$ 2.09	20.23 $\pm$ 1.36	26.12 $\pm$ 1.94	46.84 $\pm$ 1.79
2-layer GCN AE	43.04 $\pm$ 3.28	19.38 $\pm$ 3.15	23.08 $\pm$ 3.35	72.58 $\pm$ 4.54
3-layer GCN AE	44.12 $\pm$ 2.48	19.71 $\pm$ 2.55	25.94 $\pm$ 3.09	72.72 $\pm$ 1.80
Linear VAE (ours)	48.12 $\pm$ 1.96	20.71 $\pm$ 1.95	29.74 $\pm$ 0.64	49.70 $\pm$ 1.08
2-layer GCN VAE	44.84 $\pm$ 2.63	20.17 $\pm$ 3.07	25.43 $\pm$ 1.47	73.12 $\pm$ 0.83
3-layer GCN VAE	44.29 $\pm$ 2.54	19.94 $\pm$ 2.50	24.91 $\pm$ 3.09	70.56 $\pm$ 5.43