FastGAE: Fast, Scalable and Effective
Graph Autoencoders with Stochastic Subgraph Decoding

Guillaume Salha1 2  Romain Hennequin1  Jean-Baptiste Remy2
Manuel Moussallam1  Michalis Vazirgiannis2 3

Abstract

Graph autoencoders (AE) and variational autoencoders (VAE) are powerful node embedding methods, but suffer from scalability issues. In this paper, we introduce FastGAE, a general framework to scale graph AE and VAE to large graphs with millions of nodes and edges. Our strategy, based on node sampling and subgraph decoding, significantly speeds up the training of graph AE and VAE while preserving or even improving performances. We demonstrate the effectiveness of FastGAE on numerous real-world graphs, outperforming the few existing approaches to scale graph AE and VAE by a wide margin.

1. Introduction

Graph structures efficiently represent relationships and interactions among entities. Social networks, molecules, citations of scientific publications and web pages constitute some of the most notorious examples of data usually represented as graphs, i.e. as nodes connected via edges. Extracting information from these connections is essential to address numerous graph-based learning problems, ranging from link prediction to influence maximization and node clustering. In this direction, several significant improvements were recently achieved by methods leveraging node embeddings (Hamilton et al., 2017a; Wu et al., 2019b). Instead of relying on hand crafted features, these methods aim at automatically learning low-dimensional vector space representations of nodes capturing relevant information from the graph, such as structural proximity, notably through graph neural networks (Kipf & Welling, 2016b), matrix factorization (Cao et al., 2015) or random walk processes (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016).

Nonetheless, graph AE, VAE and their extensions suffer from scalability issues. As we explain in Section 2, this results from the costly decoding operations involved in the graph reconstruction. While recent works provide strategies to scale GNN models (Chen et al., 2018a; Wu et al., 2019a) i.e. encoders, the question of how to overcome complex decoders in graph AE and VAE remains open, preventing them from scaling. As a consequence, existing graph AE and VAE have been mainly applied to relatively small graphs with up to a few thousand nodes. As larger graphs are ubiquitous, we propose to address these crucial scalability concerns in this paper.

More precisely, in this paper, we introduce FastGAE, a general framework to scale graph AE and VAE to large graphs with millions of nodes and edges. We leverage effective node sampling and subgraph decoding strategies to signifi-
cantly lower the computational complexity of graph AE and VAE while preserving or even improving performances. We demonstrate the effectiveness of FastGAE on several graphs, outperforming the few existing approaches to scale graph AE and VAE by a wide margin, and we discuss and explain the performance of the proposed strategy. We publicly release the code of FastGAE\footnote{https://github.com/deezer/fastgae} for reproducibility.

This paper is organized as follows. After reviewing key notions on graph AE, VAE and their complexity in Section 2, we present our scalable framework in Section 3. We report experiments in Section 4 and we conclude in Section 5.

2. Preliminaries

In this paper, we consider an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. We denote by $A$ the binary and symmetric adjacency matrix of $G$, and by $X$ the $n \times f$ matrix stacking up $f$-dimensional node-level features vectors. For featureless graphs, we set $X = I_n$.

2.1. Graph Autoencoders

As standard autoencoders (Baldi, 2012), graph autoencoders (AE) (Tian et al., 2014; Wang et al., 2016; Kipf & Welling, 2016b) involve two stacked models.

2.1.1. Encoder

First, an encoder model aims at learning an $n \times d$ matrix $Z$, whose rows are the $d$-dimensional embedding vectors of each node $i \in \mathcal{V}$, with $d \ll n$. This matrix is usually obtained through a graph neural network (GNN) (Bruna et al., 2014; Defferrard et al., 2016; Kipf & Welling, 2016a) processing $A$ and $X$. More precisely, as detailed by Salha et al. (2020), most recent variants of graph AE implement graph convolutional networks (GCN), a model introduced by Kipf & Welling (2016a). In a L-layer GCN ($L \geq 2$), with input layer $H^{(0)} = X$ and output layer $H^{(L)} = Z$ i.e. the embedding vectors, we have:

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

$$H^{(L)} = \tilde{A}H^{(L-1)}W^{(L-1)}.$$ 

In the above equations, $\tilde{A} = D^{-1/2}(A + I_n)D^{-1/2}$ is the symmetric normalization of $A$, with $D$ the diagonal degree matrix of $A + I_n$. Also, ReLU$(x) = \max(x, 0)$ and $W^{(0)}, ..., W^{(L-1)}$ are weight matrices to tune.

2.1.2. Decoder

Then, a decoder model aims at reconstructing the graph from $Z$. Kipf & Welling (2016b) and most subsequent graph AE models implement a simple inner-product decoder. The reconstructed adjacency matrix is $\hat{A} = \sigma(ZZ^T)$ with $Z = \text{GCN}(A, X)$, and with $\sigma(\cdot)$ the sigmoid function: $\sigma(x) = 1/(1 + e^{-x})$. In other words, we have $\hat{A}_{ij} = \sigma(z_i^T z_j)$ for all $(i, j) \in \mathcal{V} \times \mathcal{V}$ i.e. nodes with larger inner products in the embedding are more likely to be connected in the graph according to the model. While we also consider this decoder in our work for simplicity and consistency with previous works, we nevertheless point out the existence of more sophisticated recent models, such as the asymmetric decoder of Salha et al. (2019b) for directed graphs, the reverse message passing schemes of Grover et al. (2019), and the decoder of Shi et al. (2020) using triadic closure.

2.1.3. Learning

As we want to build low-dimensional vector representations ensuring a good reconstruction $\hat{A}$ from the decoder, we accordingly tune the GCN weights by gradient descent (Goodfellow et al., 2016) to iteratively minimize a reconstruction loss capturing the similarity between $A$ and $\hat{A}$. In the graph AE framework, this loss is formulated as a standard cross entropy loss, where the terms with $A_{ij} = 1$ are usually re-weighted for sparse $A$ (Kipf & Welling, 2016b).

2.2. Graph Variational Autoencoders

Kipf & Welling (2016b) also introduced graph extensions of variational autoencoders (VAE) (Kingma & Welling, 2014).

2.2.1. Encoder

Graph variational autoencoders elaborate a probabilistic model on $A$, involving a $d$-dimensional latent variable $z_i$ for each node $i \in \mathcal{V}$, that corresponds to its embedding vector. Kipf & Welling (2016b) propose the following inference model as encoder: $q(Z|A, X) = \prod_{i=1}^n q(z_i|A, X)$, with $q(z_i|A, X)$ corresponding to a $N(\mu_i, \sigma^2_i)$ distribution. Kipf & Welling (2016b) use two GCNs to learn the Gaussian mean and variance parameters. In a nutshell, $\mu = \text{GCN}_\mu(A, X)$, with $\mu$ the matrix of mean vectors $\mu_i$; also, $\log \sigma = \text{GCN}_\sigma(A, X)$. The actual embedding vectors $z_i$ are samples drawn from these distributions.

2.2.2. Decoder

From these embedding vectors, a generative model aims at decoding $A$ using, as for graph AE, inner products with sigmoid activation. We have $A_{ij} = p(A_{ij} = 1|z_i, z_j) = \sigma(z_i^T z_j)$. Then, $p(A|Z) = \prod_{i=1}^n \prod_{j=1}^n p(A_{ij}|z_i, z_j)$.

2.2.3. Learning

Kipf & Welling (2016b) iteratively maximize a variational lower bound (ELBO) (Kingma & Welling, 2014) of the model’s likelihood by gradient descent w.r.t. GCNs weights: $\mathcal{L}^{\text{ELBO}} = \mathbb{E}_{q(Z|A, X)}[\log p(A|Z)] - D_{KL}(q(Z|A, X)||p(Z))$. 

FastGAE: Fast, Scalable and Effective Graph Autoencoders with Stochastic Subgraph Decoding
As explained in Section 2.3, GCN models (Kipf & Welling, 2016b) correspond to a standard Gaussian prior on the distribution of latent vectors (Kipf & Welling, 2016b).

2.3. On Complexity and Scalability

GCN models have become popular encoders for graph AE and VAE, thanks to their relative simplicity w.r.t. other GNN architectures (Bruna et al., 2014; Defferrard et al., 2016). Moreover, the cost of evaluating each layer of a GCN is linear w.r.t. the number of edges \( m \) (Kipf & Welling, 2016a), and can also be improved by replacing the encoder by a FastGCN (Chen et al., 2018a), by using simple graph convolutions (SGC) (Wu et al., 2019a) or stochastic strategies (Chen et al., 2018b; Ying et al., 2018; Chiang et al., 2019).

However, the inner-product decoder involves the multiplication of the dense matrices \( Z \) and \( Z^T \) at each training iteration. It suffers from a quadratic \( O(n^2) \) complexity, as alternative decoders (Grover et al., 2019; Salha et al., 2019b) that also require inner-products or euclidean distance computations. Storing entire \( n \times n \) dense matrices \( A \) can also lead to memory issues for large \( n \). As a consequence, despite the aforementioned solutions to scale GCN models (that were studied out of the graph AE and VAE settings) graph AE and VAE still suffer from (at least) a quadratic time complexity, and therefore from scalability issues.

As a result, existing graph AE and VAE are usually applied to relatively small graphs with up to a few thousand nodes and edges. Kipf & Welling (2016b), Grover et al. (2019) and Salha et al. (2020) only very briefly discuss random subsampling strategies as extensions for scalability. In a wider analysis, Salha et al. (2019a) propose to speed up computations by training the AE/VAE only on the smaller \( k \)-core version of the graph, then by propagating embedding representations to other nodes via faster heuristics. However, models performances tend to deteriorate for smaller cores i.e. for faster models (we compare to their approach in Section 4). To sum up, the question of how to effectively scale graph AE and VAE remains unsatisfactorily addressed.

3. Scalable Graph AE/VAE with FastGAE

In this section, we introduce our proposed framework to scale graph AE and VAE. We refer to it as FastGAE, and as variational FastGAE when applied to graph VAE.

3.1. Encoding the Entire Graph...

As explained in Section 2.3, GCN models (Kipf & Welling, 2016a) and their extensions such as FastGCN (Chen et al., 2018a) and SGC (Wu et al., 2019a) can effectively process large graphs. Therefore, in our FastGAE framework, we rely on these models to encode all the nodes into the embedding space. More precisely, in the following experiments, we implement GCN encoders for the sake of simplicity and for an easier comparison with previous works.

3.2. ...But Decoding Stochastic Subgraphs

However, while deriving node embeddings through a forward GCN pass is fast, tuning the weights of this encoder in the graph AE and VAE settings requires the computation of the aforementioned reconstruction of the entire matrix \( A \), which is intractable for large graphs.

3.2.1. Subgraph Decoding

To overcome this issue and speed up decoding, we propose to approximate reconstruction losses, by computing their values on subparts of the original graph, randomly drawn during training. More precisely, at each training iteration, we aim at decoding a random subgraph of \( \mathcal{G} \) with only \( n(S) \) nodes, with \( n(S) < n \) being a fixed parameter. Let \( \mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S) \) be such sampled subgraph, with \( \mathcal{V}_S \subset \mathcal{V} \), \( |\mathcal{V}_S| = n(S) \), and with \( \mathcal{E}_S \) denoting the subset of edges connecting the nodes in \( \mathcal{V}_S \). Instead of reconstructing the \( n \times n \) matrix \( A \), we propose to reconstruct the smaller \( n(S) \times n(S) \) matrix \( A^{(S)} \) with \( A^{(S)}_{ij} = \sigma(z_i^T z_j) \), \( \forall(i, j) \in \mathcal{V}_S \times \mathcal{V}_S \), and to only learn from the quality of the reconstructed \( A^{(S)} \) w.r.t. its ground truth counterpart \( A^{(S)} \). We draw a different subgraph \( \mathcal{G}_S \) at each training iteration, using the sampling methods detailed in the following subsections.

3.2.2. Uniform Node Sampling

A very simple way to obtain such subgraphs consists in uniformly sampling \( n(S) \) nodes from the set \( \mathcal{V} \) at each training iteration. In our experiments, we provide an implementation of this simple uniform node sampling scheme, as well as an empirical comparison to more refined strategies (see below) leveraging the graph structure for more effective sampling.

3.2.3. Node Sampling with Graph Mining

We consider alternative sampling methods, aiming at increasing the probability of including some particular nodes in the drawn subgraph w.r.t. some others. Let \( f : \mathcal{V} \rightarrow \mathbb{R}^+ \) denote a measure of the importance of nodes in the graph, obtained through graph mining methods. Assuming such function is available, we draw inspiration from word sampling in natural language processing (Mikolov et al., 2013; Goldberg & Levy, 2014) and propose to set the probability to pick each node \( i \in \mathcal{V} \) as the first element of \( \mathcal{V}(S) \) as:

\[
p_i = \frac{f(i)^\alpha}{\sum_{j \in \mathcal{V}} (f(j)^\alpha)},
\]

with \( \alpha \in \mathbb{R}^+ \). Then, assuming we sample \( n(S) \) distinct nodes without replacement, each remaining node
i ∈ V \ V(S) has a probability \( p_i/\sum_{j\notin V(S)} p_j \) to be picked as the second element of \( V(S) \), and so on until |\( V(S) \)| = \( n_S \). The previous division is a simple normalization to ensure \( \sum_{j\notin V(S)} p_j = 1 \) at each sampling step. Alternatively, one could also sample \( n_S \) nodes with replacement: it simplifies computations, as sampling probabilities are then independent of previous draws and remain fixed to \( p_i \), but a node could then be drawn several times. We stress out that, in our implementation, both variants return very similar results.

In a nutshell, important nodes according to \( f \) are more likely to be selected for decoding, and the hyperparameter \( \alpha \) helps sharpening (for \( \alpha > 1 \)) or smoothing (for \( \alpha < 1 \)) the distribution. Setting \( \alpha = 0 \) leads to the uniform node sampling of Section 3.2.2. In our experiments, we consider two importance measures \( f \) from graph mining:

- the degree of each node, which is simply the number of connections of each node: \( f(i) = \sum_{j \in V} A_{ij} \).
- the core number of each node: \( f(i) = C(i) \). The \( k \)-core version of a graph is its largest subgraph for which every node has a degree higher or equal to \( k \) within this subgraph. The core number \( C(i) \) of a node \( i \) corresponds to the largest value of \( k \) for which \( i \) is in the \( k \)-core. Core decomposition has been widely used over the past years to quantify the significance of nodes and extract representative subgraphs (see Malliaros et al. (2019) for a review). They constitute a more global importance measure than the local node degree.

Besides their popularity, we also selected these two metrics for computational efficiency. Indeed, contrary to numerous influence maximization or centrality-based measures (Newman, 2010), both can be evaluated in a linear \( O(m) \) running time (Batagelj & Zaversnik, 2003). As we empirically check in Section 4, this leads to fast and scalable computations of probability distributions, which is crucial for our FastGAE framework whose primary objective is scalability.

### 3.3. Learning from Subgraphs Reconstructions

After sampling, at each training iteration, we evaluate reconstruction losses only on the subgraph \( G(S) \), which involves fewer operations w.r.t. standard decoders. We use the resulting approximate loss for gradients computations and weights updates via mini-batch gradient descent. We note that effective subset selection for faster learning has already provided promising results in the machine learning community (Tonnaer, 2017; Kaushal et al., 2018; Gonzalez & Miikkulainen, 2019); however, contrary to these works, we focus on an unsupervised graph-based problem, and we position ourselves outside the active learning setting: since we rely on graph mining methods, our sampling method will remain fixed throughout learning.

More precisely, in standard implementations of graph AE/VAE, the cross entropy loss (from Section 2.1.3 on AE) and the negative of the ELBO’s expectation part (from Section 2.2.3 on VAE) are empirically derived by computing the following node pairs average at each iteration:

\[
\mathcal{L} = \frac{1}{n_S^2} \sum_{(i,j) \in V^2} \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}),
\]

with \( \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}) = -w[1(A_{ij} = 1)](A_{ij} \log(\hat{A}_{ij}) + (1 − A_{ij}) \log(1 − \hat{A}_{ij})) \). Here, \( 1(A_{ij} = 1) = 1 \) if \( A_{ij} = 1 \) and \( 0 \) otherwise, with \( w \) denoting a positive links re-weighting scalar parameter (commonly added on sparse graphs, and inversely proportional to the graph sparsity). Instead, in FastGAE we compute:

\[
\mathcal{L}^{\text{FastGAE}} = \frac{1}{n_S^2} \sum_{(i,j) \in V^2} \mathbb{1}_{(i,j) \in V^2_0} \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}),
\]

with \( \mathbb{1}_{(i,j) \in V^2_0} = 1 \) if the pair \((i,j) \in V^2_0\) and \( 0 \) otherwise. For variational FastGAE, we need to subtract the Kullback-Leibler divergence, as in the ELBO of standard graph VAE, to obtain our actual loss evaluation.

We stress out that, due to our degree/core-based node sampling strategies, some node pairs are more likely to appear in the subgraph than others. As a consequence, this new approximate loss \( \mathcal{L}^{\text{FastGAE}} \) is biased w.r.t. standard graph AE and VAE losses, i.e. \( \mathbb{E}(\mathcal{L}^{\text{FastGAE}}) \neq \mathcal{L} \) in general. For completeness, in the following propositions, we fully explicit the expected loss \( \mathbb{E}(\mathcal{L}^{\text{FastGAE}}) \) that we actually stochastically optimize in the FastGAE framework, as well as the formal probabilities to sample given nodes pairs. We consider both variants with and without replacement for this analysis, as the former significantly simplifies results w.r.t. the later.

We will show in the upcoming experiments that, despite this bias, optimizing this alternative loss does not deteriorate performance. On the contrary, we will provide insights exhibiting the fact that re-weighting node pairs from high degree/core nodes is actually beneficial in practice.

#### Proposition 1

Let \( G(S) = (V(S), E(S)) \) be a subgraph of \( G \) obtained from sampling \( n_S \) nodes with replacement using the node sampling strategy of FastGAE. Let \( i \) and \( j \) denote two distinct nodes from the original graph \( G \). Then:

\[
P\left(i \in V(S)\right) = 1 - (1 - p_i)^{n_S}.
\]

Also,

\[
P\left((i, j) \in V^2(S)\right) = 1 - \left[ (1 - p_i)^{n_S} + (1 - p_j)^{n_S} \right] - (1 - p_i - p_j)^{n_S}.
\]

#### Proposition 2

Let \( G(S) = (V(S), E(S)) \) be a subgraph of \( G \) obtained from sampling \( n_S \) nodes without replacement using the node sampling strategy of FastGAE. Let \( i \) and \( j \)
Proposition 3 Using the expressions of Proposition 1 (with replacement) or Proposition 2 (without replacement):

$$
E[L_{\text{FastGAE}}] = \frac{1}{n(S)} \sum_{(i,j) \in V^2} P((i,j) \in V^2(S)) L_{ij}(A_{ij}, \hat{A}_{ij}).
$$

3.4. On Complexity and Scalability

As previously mentioned, both the encoder and the sampling step of FastGAE have a linear time complexity w.r.t. $m$. Moreover, our decoder runs in $O(dn(S)^2)$ time. We point out that $n(S)$ is small in practice: in our experiments (Section 4), on large graphs, setting $n(S) \approx n/100$ consistently returns competitive performances w.r.t. baselines. Setting $n(S) \approx \sqrt{m}$, thus ensuring a $O(m)$ time complexity for FastGAE, also returns very satisfying results. Last, we do not need to store the entire dense $n \times n$ matrix $A$, but only its smaller counterpart $\hat{A}(S)$. Therefore, as we empirically verify in the next section, our proposed framework is significantly faster and more scalable than standard graph AE and VAE.

4. Empirical Analysis

4.1. Experimental Setting

4.1.1. Evaluation Tasks

As Kipf & Welling (2016b), we first consider link prediction for evaluation. We train all models on masked 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 sampled unconnected node pairs. Using decoder predictions $\hat{A}_{ij}$, we evaluate the model’s performance at classifying edges from non-edges, using the mean Area Under the ROC Curve (AUC) and Average Precision (AP) scores on test sets. For datasets with ground truth communities (see below), we also perform node clustering experiments. After training models on complete graphs, we run $k$-means algorithms in embedding spaces to cluster the $z_i$ vectors. We compare these clusters to the ground truth ones using the mean adjusted Mutual Information (MI) scores on test sets.

4.1.2. Datasets

We provide experiments on seven graphs of increasing size, whose statistics are presented in Table 1. We first study the Cora, Citeseer and Pubmed citation networks, with and without node features corresponding to $f$-dimensional bag-of-words vectors (with $f = 1, 433, 3,703$ and 500 respectively). Nodes are clustered in respectively 6, 7 and 3 topic classes, acting as ground truth communities. These datasets are common benchmarks for evaluating graph AE and VAE (Kipf & Welling, 2016b). Thanks to their relatively small size, we can directly compare the performance of FastGAE to standard graph AE and VAE on these graphs.

Then, we consider four significantly larger graphs with up to millions of nodes and edges: the Google hyperlinks web graph, the Youtube social network, the US Patent citation network, and a synthetic graph, denoted SBM, generated from a stochastic block model (see details in annex). In this last graph, by design, nodes are clustered in 100 groups of 1,000 nodes, acting as ground truth communities.

4.1.3. Models

All graph AE and VAE, with or without our FastGAE framework, were trained for 200 iterations (resp. 300) on graphs with $n < 100,000$ (resp. $n \geq 100,000$). We thoroughly checked the convergence of all models. We implemented 2-layer GCN encoders, without dropout, with 32-dim hid-

---

Table 1. Datasets Statistics

| Dataset  | Number of nodes | Number of edges |
|----------|-----------------|----------------|
| Cora     | 2 708           | 5 429          |
| Citeseer | 3 327           | 4 732          |
| Pubmed   | 19 717          | 44 338         |
| SBM      | 100 000         | 1 498 844      |
| Google   | 875 713         | 4 322 051      |
| Youtube  | 3 223 589       | 9 375 374      |
| Patent   | 3 774 768       | 16 518 948     |

---

2https://linqs.soe.ucsc.edu/data

3http://snap.stanford.edu/data/index.html

4http://konect.uni-koblenz.de/networks/
4.2. Preliminary Insights: Learning from High Degree and Core Nodes is Crucial for Good Performances

As a preliminary to our results on FastGAE, we report important insights from experiments on standard graph AE/VAE. They motivated the design of our framework and emphasize the relevance of sampling high-degree/core nodes.

On the medium-size Cora, Citeseer and Pubmed graphs, we trained standard graph AE and VAE models (on entire graphs), but tried to mask \( k \) nodes and their edges from the decoder i.e. from the loss computations, for different values of \( k \). Such masking procedure is expected to lower performances, as the model leverages less information about the quality of the reconstruction for learning.

Figure 1 shows that, when these \( k \) removed nodes are the top-\( k \) highest degrees/cores nodes, performances on the link prediction task tumble down. On the contrary, removing the \( k \) nodes with minimal degrees or core numbers from the loss leads to almost no drop (and even slightly better results on Pubmed, which suggests that removing non-informative nodes might even be beneficial for learning). In supplementary material, we report similar results on MI scores for node clustering. Therefore, when implementing stochastic subgraph decoding strategies for scalability, sampling high-degree/core nodes is crucial to build effective embeddings. FastGAE, which explicitly exploits these structural node properties, and re-weights the corresponding node pairs in expected losses, is consistent with such insight.

4.3. Experiments on FastGAE

In the remaining of this section, we provide an empirical evaluation of FastGAE and its variational FastGAE variant.

4.3.1. MEDIUM-SIZE GRAPHS

For medium-size graphs, we can directly compare our framework to standard graph AE and VAE models. Table 2 details mean AUC and AP scores and their standard errors over 100 runs with different train/test splits for the link prediction task on Pubmed with AE models. For sample sizes \( n(S) = 5\,000, 2\,000 \) and \( 1\,000 \), i.e. up to 20 times smaller than \( n \), our FastGAE framework with degree and core sampling both achieve competitive or even outperforming results w.r.t. standard graph AE (e.g. +2.31 AUC points for FastGAE with degree sampling and \( n(S) = 5\,000 \)).
Table 2. Link prediction on Pubmed ($n = 19,717, m = 44,338$) using standard Graph AE, FastGAE, and baselines

| Model          | Subgraphs size $n_i$ | Average Performance on Test Set | Average Running Times (in seconds) |
|----------------|----------------------|----------------------------------|-----------------------------------|
|                |                      | AUC (in %) | AP (in %) | Compute $p_i$ | Train model | Total | Speed gain w.r.t. Graph AE |
| Standard Graph AE | -                    | 82.51 ± 0.64 | 87.42 ± 0.38 | -              | 811.43      | 811.43 | -                          |
| FastGAE with degree sampling ($α^* = 1$) | 5 000 | 84.82 ± 0.32 | 88.19 ± 0.23 | 0.01 | 14.41 | 14.42 | × 56.27 |
|                | 2 500 | 84.12 ± 0.40 | 87.56 ± 0.30 | 0.01 | 5.72 | 5.73 | × 141.61 |
|                | 1 000 | 83.53 ± 0.41 | 86.12 ± 0.32 | 0.01 | 3.14 | 3.15 | × 257.60 |
|                | 500  | 82.68 ± 0.51 | 85.89 ± 0.47 | 0.01 | 2.98 | 2.99 | × 271.38 |
|                | 250  | 80.77 ± 0.55 | 84.05 ± 0.51 | 0.01 | 2.83 | 2.84 | × 285.71 |
| FastGAE with core sampling ($α^* = 2$) | 5 000 | 84.62 ± 0.24 | 88.09 ± 0.16 | 1.75 | 15.98 | 15.98 | × 45.77 |
|                | 2 500 | 83.69 ± 0.34 | 87.28 ± 0.31 | 1.75 | 7.51 | 7.51 | × 87.63 |
|                | 1 000 | 82.50 ± 0.44 | 86.33 ± 0.43 | 1.75 | 4.75 | 4.75 | × 128.44 |
|                | 500  | 80.96 ± 0.52 | 84.86 ± 0.46 | 1.75 | 4.57 | 4.57 | × 128.39 |
|                | 250  | 79.53 ± 0.53 | 83.10 ± 0.50 | 1.75 | 4.44 | 4.44 | × 131.04 |
| FastGAE with uniform sampling | 5 000 | 81.08 ± 0.48 | 85.90 ± 0.60 | - | 13.90 | 13.90 | × 38.37 |
|                | 2 500 | 78.72 ± 0.74 | 83.50 ± 0.75 | - | 5.48 | 5.48 | × 148.07 |
|                | 1 000 | 76.63 ± 0.82 | 80.98 ± 0.62 | - | 3.06 | 3.06 | × 265.17 |
|                | 500  | 75.09 ± 2.05 | 78.53 ± 2.04 | - | 2.98 | 2.98 | × 271.29 |
|                | 250  | 74.12 ± 2.07 | 77.72 ± 1.22 | - | 2.82 | 2.82 | × 287.74 |
| Best Core-Graph AE ($k = 2$) | - | 84.30 ± 0.27 | 86.11 ± 0.43 | - | 168.91 | 168.91 | × 4.80 |
| Fastest Core Graph AE ($k = 9$) | - | 61.65 ± 0.94 | 64.82 ± 0.72 | - | 2.92 | 2.92 | × 277.89 |
| node2vec | - | 81.25 ± 0.26 | 83.55 ± 0.26 | - | 48.91 | 48.91 | × 16.59 |
| Spectral Embedding | - | 83.14 ± 0.42 | 86.55 ± 0.41 | - | 31.71 | 31.71 | × 25.59 |

In supplementary material, we consolidate our results by reaching similar conclusions on VAE, on the node clustering task, and on the variant of Pubmed with node-level features. We also report similar results on Cora and Citeseer, with and without features, for these two tasks, and we provide details on all optimal values for the hyperparameter $α$ (denoted $α^*$ in our tables), used for degree and core sampling.

Last, to complete our analysis, in all these settings, we also compare FastGAE to a spectral embedding (embedding axes are eigenvectors of $G$’s Laplacian matrix), which is a powerful but not scalable baseline, to node2vec (Grover & Leskovec, 2016), another very popular and scalable node embedding method (we report hyperparameters in supplementary material) and, for node clustering, to Louvain’s scalable community detection algorithm (Blondel et al., 2008). FastGAE and variational FastGAE almost always reach competitive and faster results w.r.t. these baselines.

4.3.2. Large Graphs

Regarding large graphs, Table 3 details mean MI scores for node clustering experiments on the SBM graph with VAE models, and Table 4 provides mean AUC and AP scores for link prediction on the Patent graph with AE models. We report more experiments on these graphs and on Google and Youtube in supplementary material. All scores are averaged over 10 runs with different train/test splits. On large graphs, direct comparison with standard graph AE and VAE is impossible, as these models are intractable. However, our FastGAE and variational FastGAE almost always reach competitive or better results w.r.t. the Louvain and node2vec baselines, which emphasizes the representational power of...
graph AE/VAE on large graphs. FastGAE-based models are also significantly faster. Moreover, as for medium-size graphs, core and degree sampling achieve better results than uniform sampling (e.g., +6.94 AUC points for FastGAE with degree sampling on Patent, with $n(S) = 20 000$). We notice that computing the $p_i$ probabilities through core sampling is much longer on large graphs, but bring no empirical benefit w.r.t. degree sampling: we therefore recommend using degree sampling for large graphs.

Last, core-based and, especially, degree-based FastGAE also both provide better performance/speed trade-offs than Core-Graph AE/VAE. We point out that Core-Graph AE/VAE models are intractable on the SBM dataset due to the lack of size decreasing core structure on this graph (the 21-core of SBM includes 95 200 nodes, which is too large to train a graph AE or VAE on our machines, and the 22-core is empty). On such particular graph structure, FastGAE appears as the unique option to scale graph AE and VAE.

### 4.3.3. Extensions

FastGAE is a flexible framework that easily extends to alternative encoders, such as FastGCN (Chen et al., 2018a), and decoders, e.g. to include adversarial training (Pan et al., 2018), or to apply FastGAE to directed graphs (Salha et al., 2019b). However, in this paper we always assume that the graph is fixed. Future works will consider extensions of FastGAE for scalable dynamic graph embeddings.

### 5. Conclusion

We introduced and released a general framework to scale graph AE and VAE models. We demonstrated its effectiveness on large graphs with up to millions of nodes and edges, both in terms of speed, of scalability and of performance.
References

Abbe, E. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

Baldi, P. Autoencoders, unsupervised learning, and deep architectures. *ICML Workshop on Unsupervised and Transfer Learning*, 2012.

Batagelj, V. and Zaversnik, M. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

Berg, R. v. d., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. *KDD Deep Learning Day*, 2018.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiments*, 2008(10):P10008, 2008.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations (ICLR)*, 2014.

Cao, S., Lu, W., and Xu, Q. Grarep: Learning graph representations with global structural information. *ACM International Conference on Information and Knowledge Management (CIKM)*, 2015.

Chen, J., Ma, T., and Xiao, C. Fastgcn: fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations (ICLR)*, 2018a.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. *International Conference on Machine Learning (ICML)*, 2018b.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Goldberg, Y. and Levy, O. word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Gonzalez, S. and Miikkulainen, R. Improved training speed, accuracy, and data utilization through loss function optimization. *arXiv preprint arXiv:1905.11528*, 2019.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Grover, A., Zweig, A., and Ermon, S. Graphite: Iterative generative modeling of graphs. *International Conference on Machine Learning (ICML)*, 2019.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017a.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017b.

Hasanazadeh, A., Hajiramezanali, E., Narayanan, K., Duffield, N., Zhou, M., and Qian, X. Semi-implicit graph variational auto-encoders. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning (ICML)*, 2018.

Kaushal, V., Sahoo, A., Doctor, K., Raju, N., Shetty, S., Singh, P., Iyer, R., and Ramakrishnan, G. Learning from less data: Diversified subset selection and active learning in image classification tasks. *arXiv preprint arXiv:1805.11191*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016b.

Kleinberg, B., Li, Y., and Yuan, Y. An alternative view: When does sgd escape local minima? *International Conference on Machine Learning (ICML)*, 2018.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
Ma, T., Chen, J., and Xiao, C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Malliaros, F. D., Giatsidis, C., Papadopoulos, A. N., and Vazirgiannis, M. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, pp. 1–32, 2019.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Newman, M. *Networks: An Introduction*. UK Oxford University Press, 2010.

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

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. *Parallel Distributed Processing, Vol 1*, 1986.

Salha, G., Hennequin, R., Tran, V. A., and Vazirgiannis, M. A degeneracy framework for scalable graph autoencoders. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019a.

Salha, G., Limnios, S., Hennequin, R., Tran, V. A., and Vazirgiannis, M. Gravity-inspired graph autoencoders for directed link prediction. *ACM International Conference on Information and Knowledge Management (CIKM)*, 2019b.

Salha, G., Hennequin, R., and Vazirgiannis, M. Simple and effective graph autoencoders with one-hop linear models. *arXiv preprint arXiv:2001.07614*, 2020.

Samanta, B., Abir, D., Jana, G., Chattaraj, P. K., Ganguly, N., and Rodriguez, M. G. Nevae: A deep generative model for molecular graphs. *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Shi, H., Fan, H., and Kwok, J. T. Effective decoding in graph auto-encoder using triadic closure. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. *International Conference on Artificial Neural Networks (ICANN)*, 2018.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. *International Conference on World Wide Web (WWW)*, 2015.

Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. Learning deep representations for graph clustering. *AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

Tonnaer, L. Active learning in vae latent space. *Eindhoven University of Technology*, 2017.

Tran, P. V. Multi-task graph autoencoders. *arXiv preprint arXiv:1811.02798*, 2018.

Wang, C., Pan, S., Long, G., Zhu, X., and Jiang, J. Mgae: Marginalized graph autoencoder for graph clustering. *ACM Conference on Information and Knowledge Management (CIKM)*, 2017.

Wang, D., Cui, P., and Zhu, W. Structural deep network embedding. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. *International Conference on Machine Learning (ICML)*, 2019a.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019b.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.
Supplementary Material

This supplementary material provides proofs and complementary experiments for the paper FastGAE: Fast, Scalable and Effective Graph Autoencoders with Stochastic Subgraph Decoding.

Proof of Proposition 1

Let $\mathcal{G}(S) = (\mathcal{V}(S), \mathcal{E}(S))$ be a subgraph of $\mathcal{G}$ obtained from sampling $n(S)$ nodes with replacement using the node sampling strategy of FastGAE. In this setting, the probability to draw a node $i \in \mathcal{V}$ depends on nodes previously drawn.

Let $G$ be the original graph.

Indeed, for $i$ not to belong to $\mathcal{V}(S)$, it must not be selected at any of the $n(S)$ draws, which happens with probability $1 - p_i$ for each draw. Therefore:

$$P(i \notin \mathcal{V}(S)) = (1 - p_i)^{n(S)}.$$  \hspace{1cm} (1)

Moreover, let $i$ and $j$ denote two distinct nodes from the original graph $\mathcal{G}$. We have:

$$P((i, j) \notin \mathcal{V}^2(S)) = P(i \notin \mathcal{V}(S) \text{ or } j \notin \mathcal{V}(S)) = P(i \notin \mathcal{V}(S)) + P(j \notin \mathcal{V}(S)) - P(i \notin \mathcal{V}(S), j \notin \mathcal{V}(S))$$

with, using the previous result from (1), $P(i \notin \mathcal{V}(S)) = (1 - p_i)^{n(S)}$ and $P(j \notin \mathcal{V}(S)) = (1 - p_j)^{n(S)}$. Using a similar argument, we also obtain:

$$P(i \notin \mathcal{V}(S), j \notin \mathcal{V}(S)) = (1 - (p_i + p_j))^{n(S)}.$$  \hspace{1cm} (2)

Therefore:

$$P((i, j) \notin \mathcal{V}^2(S)) = \left[ (1 - p_i)^{n(S)} + (1 - p_j)^{n(S)} - (1 - p_i - p_j)^{n(S)} \right].$$

And:

$$P((i, j) \in \mathcal{V}^2(S)) = 1 - P((i, j) \notin \mathcal{V}^2(S))$$

$$= 1 - \left[ (1 - p_i)^{n(S)} + (1 - p_j)^{n(S)} - (1 - p_i - p_j)^{n(S)} \right].$$  \hspace{1cm} (3)

Last, for self-loops:

$$P((i, i) \in \mathcal{V}^2(S)) = P(i \in \mathcal{V}(S)) = 1 - (1 - p_i)^{n(S)}.$$  \hspace{1cm} (4)

Proof of Proposition 2

Let $\mathcal{G}(S) = (\mathcal{V}(S), \mathcal{E}(S))$ be a subgraph of $\mathcal{G}$ obtained from sampling $n(S)$ nodes without replacement using the node sampling strategy of FastGAE. In this setting, the probability to draw a node $i \in \mathcal{V}$ depends on nodes previously drawn.

Let:

$$U(i) = \{u \subset \mathcal{V}, |u| = n(S) \text{ and } i \in u \}$$

denote the set of of all ordered subsets of $n(S)$ distinct nodes that include node $i$. With such notations:

$$P(i \in \mathcal{V}(S)) = \sum_{U \in U(i)} P(\mathcal{V}(S) = U).$$

For a given set $U \in U(i)$, let us denote by $(u_1, u_2, ..., u_{n(S)})$ its ordered elements. Also, let $(\mathcal{V}(S)_1, \mathcal{V}(S)_2, ..., \mathcal{V}(S)_{n(S)})$ denote two distinct nodes from the ordered nodes of set $\mathcal{G}(S)$ (i.e. $\mathcal{V}(S)_1$ is the first drawn node, $\mathcal{V}(S)_2$ is the second one, etc). We have:

$$P(\mathcal{V}(S) = U) = P(\mathcal{V}(S)_1 = u_1, \mathcal{V}(S)_2 = u_2, ..., \mathcal{V}(S)_{n(S)} = u_{n(S)})$$

$$= P(\mathcal{V}(S)_1 = u_1) \prod_{k=2}^{n(S)} P(\mathcal{V}(S)_k = u_k | \mathcal{V}(S)_{k-1} = u_{k-1}, ..., \mathcal{V}(S)_{1} = u_1)$$

$$= p_{u_1} \prod_{k=2}^{n(S)} \frac{p_{u_k}}{1 - \sum_{k'=1}^{k-1} p_{u_{k'}}}.$$  \hspace{1cm} (5)

Therefore:

$$P(i \in \mathcal{V}(S)) = \sum_{U \in U(i)} p_{u_1} \prod_{k=2}^{n(S)} \frac{p_{u_k}}{1 - \sum_{k'=1}^{k-1} p_{u_{k'}}}. \hspace{1cm} (6)$$

Moreover, let $i$ and $j$ denote two distinct nodes from the original graph $\mathcal{G}$. Using similar notations and reasoning, we get:

$$P((i, j) \in \mathcal{V}^2(S)) = \sum_{U \in U(i) \cap U(j)} P(\mathcal{V}(S) = U).$$

Therefore:

$$P((i, j) \in \mathcal{V}^2(S)) = \sum_{U \in U(i) \cap U(j)} p_{u_1} \prod_{k=2}^{n(S)} \frac{p_{u_k}}{1 - \sum_{k'=1}^{k-1} p_{u_{k'}}}. \hspace{1cm} (7)$$

And, for self-loops, $P((i, i) \in \mathcal{V}^2(S)) = P(i \in \mathcal{V}(S))$. 

Proof of Proposition 3

We have:

\[ \mathbb{E}[L_{\text{FastGAE}}] = \mathbb{E} \left[ \frac{1}{n^2(S)} \sum_{(i,j) \in V^2} \mathbf{1}_{((i,j) \in V^2(S))} \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}) \right] \]

\[ = \frac{1}{n^2(S)} \sum_{(i,j) \in V^2} \mathbb{E} \left[ \mathbf{1}_{((i,j) \in V^2(S))} \right] \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}) \]

\[ = \frac{1}{n^2(S)} \sum_{(i,j) \in V^2} \mathbb{P} \left((i,j) \in V^2(S)\right) \mathcal{L}_{ij}(A_{ij}, \hat{A}_{ij}). \]

By replacing \( \mathbb{P}((i,j) \in V^2(S)) \) by the expressions of Proposition 1 (with replacement) or Proposition 2 (without replacement), we obtain an explicit formulation for \( \mathbb{E}[L_{\text{FastGAE}}] \).

Complementary experiments

The following tables and figures present complementary experimental results to support our evaluation of FastGAE:

- Table 5 reports complementary experiments on all graph datasets, for the *link prediction* task.
- Table 6 reports complementary experiments on graph datasets with ground truth communities, for the *node clustering* task.
- Figure 2 provides similar node masking experiments w.r.t. Figure 1 from main paper, but displays the mean adjusted Mutual Information (MI) scores instead of the mean AUC. These experiments confirm the insights from Section 4.2., i.e. that learning from high degree and core nodes is crucial for good performances.
- Figure 3 details the optimal values of the hyperparameter \( \alpha \) from our node sampling scheme, for all graphs and for both core-based and degree-based sampling.

In these experiments and in the main paper, we trained *node2vec* models with hyperparameters \( p = 1 \) and \( q = 1 \), from 10 random walks of length 80 per node, with a window size of 5 and on a single epoch. Also, the *Core-Graph AE/VAE* baseline refers to the alternative framework for scalable graph autoencoders introduced by Salha et al. (2019a) with optimal values (regarding mean AUC scores) for the hyperparameter \( k \) detailed in tables. Last, our *SBM graph* is a synthetic graph generated from a stochastic block model (Abbe, 2017), which is a generative model for random graphs. Nodes are clustered in 100 ground truth communities of 1 000 nodes. Two nodes from a same community (resp. from different communities) are connected by an edge with probability \( 2 \times 10^{-2} \) (resp. \( 2 \times 10^{-4} \)). For reproducibility, we release this generated graph with our source code.
Table 5. Complementary link prediction experiments. For each graph, for brevity, we only report the best graph AE or VAE model in terms of AUC and AP scores, the best degree-based FastGAE version of this model, and the best baseline (among Core-Graph AE/VAE, node2vec and the spectral embedding). FastGAE always reaches competitive and faster results w.r.t. standard AE/VAE and baselines. FastGAE can provide faster results than those reported, by decreasing \( n(S) \) and losing a few AUC/AP points. Scores are averaged over 100 runs (resp. 10 runs) for medium-size graphs (resp. for large graphs).

| Dataset   | Model                                      | Average Performance on Test Set | Average Running Times (in seconds) | Speed Gain w.r.t. Graph AE/VAE |
|-----------|--------------------------------------------|---------------------------------|----------------------------------|--------------------------------|
|           |                                            | AUC (in %)                      | AP (in %)                        | Compute \( p_i \) | Train model | Total |                    |
| Cora      | Standard Graph AE                          | 84.79 ± 1.10                   | 88.45 ± 0.82                     | 0.002             | 1.62       | 1.622 | × 2.39          |
|           | FastGAE with degree sampling (\( \alpha = 2, n(S) = 1000 \)) | 84.64 ± 1.18                   | 87.75 ± 1.14                     |                    |            |       |                    |
|           | Best baseline: Spectral Embedding          | 86.49 ± 0.98                   | 87.42 ± 1.04                     |                    |            |       |                    |
| Cora with features | Standard Graph VAE                      | 91.64 ± 0.92                   | 92.66 ± 0.91                     |                    |            |       |                    |
|           | Variational FastGAE with degree sampling (\( \alpha = 2, n(S) = 500 \)) | 91.43 ± 0.95                   | 92.55 ± 1.18                     |                    |            |       |                    |
|           | Best baseline: Core-Graph VAE (\( k = 2 \)) | 87.94 ± 1.12                   | 89.00 ± 1.11                     |                    |            |       |                    |
| Citeseer  | Standard Graph AE                          | 78.25 ± 1.69                   | 83.79 ± 1.24                     |                    |            |       |                    |
|           | FastGAE with degree sampling (\( \alpha = 1, n(S) = 1000 \)) | 78.31 ± 1.25                   | 82.40 ± 0.99                     |                    |            |       |                    |
|           | Best baseline: Spectral Embedding          | 80.42 ± 1.38                   | 83.75 ± 1.12                     |                    |            |       |                    |
| Citeseer with features | Standard Graph VAE                      | 90.72 ± 1.01                   | 92.05 ± 0.97                     |                    |            |       |                    |
|           | Variational FastGAE with degree sampling (\( \alpha = 1, n(S) = 500 \)) | 90.09 ± 1.08                   | 90.23 ± 0.88                     |                    |            |       |                    |
|           | Best baseline: Core-Graph VAE (\( k = 2 \)) | 81.85 ± 1.72                   | 83.65 ± 1.64                     |                    |            |       |                    |
| Pubmed    | Standard Graph AE                          | 82.51 ± 0.64                   | 87.42 ± 0.38                     |                    |            |       |                    |
|           | FastGAE with degree sampling (\( \alpha = 1, n(S) = 5000 \)) | 84.62 ± 0.32                   | 88.19 ± 0.23                     | 0.01              | 14.41      | 14.42 | × 56.27          |
|           | Best baseline: Core-Graph AE (\( k = 2 \)) | 84.30 ± 0.27                   | 86.11 ± 0.43                     |                    |            |       |                    |
| Pubmed with features | Standard Graph AE                      | 96.28 ± 0.36                   | 96.29 ± 0.25                     |                    |            |       |                    |
|           | FastGAE with degree sampling (\( \alpha = 1, n(S) = 5000 \)) | 96.12 ± 0.20                   | 96.35 ± 0.19                     | 0.01              | 18.74      | 19.75 | × 48.23          |
|           | Best baseline: Core-Graph AE (\( k = 2 \)) | 85.34 ± 0.33                   | 86.06 ± 0.24                     |                    |            |       |                    |
| SBM       | Standard Graph VAE                         | (intractable)                   | (intractable)                    | 0.03              | 43.86      | 43.89 |                    |
|           | Variational FastGAE with degree sampling (\( \alpha = 2, n(S) = 500 \)) | (intractable)                   | (intractable)                    |                    |            |       |                    |
|           | Best baseline: node2vec                    | 80.89 ± 0.32                   | 83.51 ± 0.29                     |                    |            |       |                    |
| Google    | Standard Graph AE                          | (intractable)                   | (intractable)                    | 0.14              | 152.02     | 152.16 | (2min30)         |
|           | FastGAE with degree sampling (\( \alpha = 1, n(S) = 10000 \)) | (intractable)                   | (intractable)                    |                    |            |       |                    |
|           | Best baseline: node2vec                    | 94.89 ± 0.63                   | 96.82 ± 0.72                     |                    |            |       |                    |
| Youtube   | Standard Graph VAE                         | (intractable)                   | (intractable)                    | 0.28              | 3 596.03   | 3 596.31 | (1h)            |
|           | Variational FastGAE with degree sampling (\( \alpha = 5, n(S) = 20000 \)) | (intractable)                   | (intractable)                    |                    |            |       |                    |
|           | Best baseline: Core-Graph VAE (\( k = 20 \)) | 80.53 ± 0.23                   | 82.45 ± 0.20                     |                    |            |       |                    |
| Patent    | Standard Graph AE                          | (intractable)                   | (intractable)                    | 0.30              | 4 401.67   | 4 401.67 | (1h13)          |
|           | FastGAE with degree sampling (\( \alpha = 2, n(S) = 20000 \)) | (intractable)                   | (intractable)                    |                    |            |       |                    |
|           | Best baseline: node2vec                    | 92.96 ± 0.23                   | 93.43 ± 0.17                     |                    |            |       |                    |
Table 6. Node clustering on graphs with communities. For each graph, for brevity, we only report the best graph AE or VAE model in terms of Mutual Information score, the best degree-based FastGAE version of this model, and the best baseline (among Core-Graph AE/VAE, Louvain, node2vec and the spectral embedding). FastGAE always reaches competitive and faster results w.r.t. standard AE/VAE; however, we note that Louvain outperforms AE/VAE on Cora/Citeseer/SBM.

FastGAE can provide faster results than those reported, by decreasing $n_i(S)$ and losing a few MI points. Scores are averaged over 100 runs (resp. 10 runs) for medium-size graphs (resp. SBM).

| Dataset | Model | Average Performance on Test Set | Average Running Times (in seconds) | Speed Gain w.r.t. Graph AE/VAE |
|---------|-------|---------------------------------|-----------------------------------|-------------------------------|
| Cora    | Standard Graph AE               | 30.88 ± 2.56                    | -                                 | × 2.12                        |
|         | FastGAE with degree sampling    | 35.56 ± 2.80                    | -                                 | -                             |
|         | (α∗ = 2, $n_i(S) = 1,000$)     |                                 | 0.002 1.60 1.602                  | × 2.12                        |
|         | Best baseline: Louvain          | 46.72 ± 0.85                    | -                                 | × 2.03                        |
| Citeseer| Standard Graph VAE              | 44.84 ± 2.63                    | -                                 | -                             |
|         | Variational FastGAE with degree sampling | 42.78 ± 2.44 | 0.002 2.23 2.232 | × 1.94 |
|         | (α∗ = 2, $n_i(S) = 500$)       |                                 | -                                 | -                             |
|         | Best baseline: Louvain          | 46.72 ± 0.85                    | -                                 | × 2.41                        |
| Pubmed  | Standard Graph VAE              | 20.17 ± 2.07                    | -                                 | -                             |
|         | Variational FastGAE with degree sampling | 20.49 ± 3.43 | 0.002 2.83 2.832 | × 2.16 |
|         | (α∗ = 1, $n_i(S) = 500$)       |                                 | -                                 | -                             |
|         | Best baseline: Cora-Graph VAE ($k = 2$) | 16.53 ± 1.95 | - 2.76 2.76 | × 2.22 |
| SBM     | Standard Graph VAE              | 20.52 ± 2.97                    | -                                 | -                             |
|         | Variational FastGAE with degree sampling | 22.81 ± 4.80 | 0.01 13.64 13.65 | × 62.71 |
|         | (α∗ = 1, $n_i(S) = 500$)       |                                 | -                                 | -                             |
|         | Best baseline: Core-Graph VAE ($k = 2$) | 23.56 ± 3.12 | - 50.11 50.11 | × 17.08 |
|         | (intractable)                   | (intractable)                   | -                                 | -                             |

Figure 2. Node clustering on featureless Cora, Citeseer and Pubmed using standard Graph VAE models, trained while masking $k$ nodes and their connections from the decoder i.e. from loss computations. MI scores are averaged over 100 runs with random train/test splits.
Optimal values of $\alpha$ for degree and core sampling w.r.t. mean AUC scores on validation sets, for Variational FastGAE models.

Figure 3.