Appendices

A. Proof of Theorem 2

Proof. For simplicity, we state the proof for a single variational marginal embedding $\mu_i^{(l)}$ and consider that $\mu_i^{(l)}$ for all $l \in \mathbb{N} \cup 0$ are unidimensional.

Let us denote $\mathbf{N}_i^{(l)} \in \mathbb{R}^n$ to be the vector of neighboring kernel embeddings at iteration l such that the j-th entry of $\mathbf{N}_i^{(l)}$ corresponds to $\mu_j^{(l)}$ if $j \in \mathcal{N}(i)$ and zero otherwise. Hence, we can rewrite Eq. (17) as:

$$\mu_i^{(l)} = \tilde{O}_{\psi,\mathcal{G}}\left(\mathbf{N}_i^{(l-1)}\right) \tag{18}$$

where we have overloaded $\tilde{O}_{\psi,,\mathcal{G}}$ to now denote a function that takes as argument an n-dimensional vector of marginal embeddings.

Assuming that the function $\tilde{O}_{\psi,\mathcal{G}}$ is differentiable, a first-order Taylor expansion of Eq. (18) around the origin $\mathbf{0}$ is given by:

$$\mu_i^{(l)} \approx \tilde{O}_{\psi,\mathcal{G}}\left(\mathbf{0}\right) + \mathbf{N}_i^{(l-1)} \cdot \nabla \tilde{O}_{\psi,\mathcal{G}}\left(\mathbf{0}\right). \tag{19}$$

Since every marginal density is unidimensional, we now consider a GNN with a single activation per node in every layer, *i.e.*, $\mathbf{H}^{(l)} \in \mathbb{R}^n$ for all $l \in \mathbb{N} \cup 0$. This also implies that the bias can be expressed as an n-dimensional vector, *i.e.*, $\mathbf{B}_l \in \mathbb{R}^n$ and we have a single weight parameter $W_l \in \mathbb{R}$. For a single entry of $\mathbf{H}^{(l)}$, we can specify Eq. (2) component-wise as:

$$H_i^{(l)} = \eta_l\left(B_{l,i} + \sum_{f \in \mathcal{F}_l} f(\mathbf{A}_i)\mathbf{H}^{(l-1)}W_l\right) \tag{20}$$

where $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$ and is non-zero only for entries corresponding to the neighbors of node i.

Now, consider the following instantiation of Eq. (20):

- $\eta_l \leftarrow \mathcal{I}$ (identity function)
- $B_{l,i} \leftarrow \tilde{O}_{\psi,\mathcal{G}}\left(\mathbf{0}\right)$
- A family of n transformations $\mathcal{F}_l = \{f_{l,j}\}_{j=1}^n$ where $f_{l,j}(\mathbf{A}_i) = \frac{\partial \tilde{O}_{\psi,\mathcal{G}}}{\partial \mu_j^{(l-1)}}\left(\mathbf{0}\right) A_{ij}$
- $H_i^{(l-1)} \leftarrow \mu_i^{(l-1)}$
- $W_l \leftarrow 1$.

With the above substitutions, we can equate the first order approximation in Eq. (18) to the GNN message passing rule in Eq. (20), thus completing the proof. With vectorized notation and use of matrix calculus in Eqs. (18-20), the derivation above also applies to entire vectors of variational marginal embeddings with arbitrary dimensions. □

B. Experiment Specifications

B.1. Link prediction

We used the SC implementation from (Pedregosa et al., 2011) and public implementations for others made available by the authors. For SC, we used a dimension size of 128. For DeepWalk and node2vec which uses a skipgram like objective on random walks from the graph, we used the same dimension size and default settings used in (Perozzi et al., 2014) and (Grover & Leskovec, 2016) respectively of 10 random walks of length 80 per node and a context size of 10. For node2vec, we searched over the random walk bias parameters using a grid search in $\{0.25, 0.5, 1, 2, 4\}$ as prescribed in the original work. For GAE and VGAE, we used the same architecture as VGAE and Adam optimizer with learning rate of 0.01.

For Graphite-AE and Graphite-VAE, we used an architecture of 32-32 units for the encoder and 16-32-16 units for the decoder (two rounds of iterative decoding before a final inner product). The model is trained using the Adam optimizer (Kingma & Welling, 2014) with a learning rate of 0.01. All activations were RELUs.The dropout rate (for edges) and λ were tuned as hyperparameters on the validation set to optimize the AUC, whereas traditional dropout was set to 0 for all datasets. Additionally, we trained every model for 500 iterations and used the model checkpoint with the best validation loss for testing. Scores are reported as an average of 50 runs with different train/validation/test splits (with the requirement that the training graph necessarily be connected).

For Graphite, we observed that using a form of skip connections to define a linear combination of the initial embedding $\mathbf{Z}$ and the final embedding $\mathbf{Z}^*$ is particularly useful. The skip connection consists of a tunable hyperparameter λ controlling the relative weights of the embeddings. The final embedding of Graphite is a function of the initial embedding Z and the last induced embedding Z^*. We consider two functions to aggregate them into a final embedding. That is, $(1-\lambda)Z + \lambda Z^*$ and $Z + \lambda Z^* / \|Z^*\|$, which correspond to a convex combination of two embeddings, and an incremental update to the initial embedding in a given direction, respectively. Note that in either case, GAE and VGAE reduce to a special case of Graphite, using only a single inner-product decoder (*i.e.*, $\lambda = 0$). On Cora and Pubmed final embeddings were derived through convex combination, on Citeseer through incremental update.

Scalability. We experimented with learning VGAE and Graphite models by subsampling $|E|$ random entries for Monte Carlo evaluation of the objective at each iteration. The corresponding AUC scores are shown in Table 6. The results suggest that Graphite can effectively scale to large graphs without significant loss in accuracy. The AUC results

Table 6. AUC scores for link prediction with Monte Carlo subsampling during training. Higher is better.

	Cora	Citeseer	Pubmed
VGAE	89.6	92.2	92.3
Graphite	**90.5**	**92.5**	**93.1**

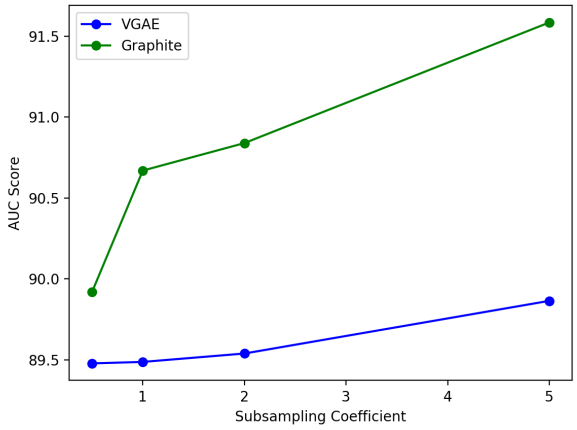

Figure 4. AUC score of VGAE and Graphite with subsampled edges on the Cora dataset.

trained with edge subsampling as we vary the subsampling coefficient c are shown in Figure 4.

B.2. Semi-supervised node classification

We report the baseline results for SemiEmb (Weston et al., 2008), DeepWalk (Perozzi et al., 2014), ICA (Lu & Getoor, 2003) and Planetoid (Yang et al., 2016) as specified in (Kipf & Welling, 2017). GCN uses a 32-16 architecture with ReLu activations and early stopping after 10 epochs without increasing validation accuracy. The Graphite model uses the same architecture as in link prediction (with no edge dropout). The parameters of the posterior distributions are concatenated with node features to predict the final output. The parameters are learned using the Adam optimizer (Kingma & Welling, 2014) with a learning rate of 0.01. All accuracies are taken as an average of 100 runs.

B.3. Density estimation

To accommodate for input graphs of different sizes, we learn a model architecture specified for the maximum possible nodes (*i.e.*, 20 in this case). While feeding in smaller graphs, we simply add dummy nodes disconnected from the rest of the graph. The dummy nodes have no influence on the gradient updates for the parameters affecting the latent or observed variables involving nodes in the true graph. For the experiments on density estimation, we pick a graph family, then train and validate on graphs sampled exclusively from that family. We consider graphs with nodes ranging between 10 and 20 nodes belonging to the following graph families :

- Erdos-Renyi (Erdös & Rényi, 1959): each edge independently sampled with probability $p = 0.5$
- Ego Network: a random Erdos-Renyi graph with all nodes neighbors of one randomly chosen node
- Random Regular: uniformly random regular graph with degree $d = 4$
- Random Geometric: graph induced by uniformly random points in unit square with edges between points at euclidean distance less than $r = 0.5$
- Random Power Tree: Tree generated by randomly swapping elements from a degree distribution to satisfy a power law distribution for $\gamma = 3$
- Barabasi-Albert (Barabasi & Albert, 1999): Preferential attachment graph generation with attachment edge count $m = 4$

We use convex combinations over three successively induced embeddings. Scores are reported over an average of 50 runs. Additionally, a two-layer neural net is applied to the initially sampled embedding $\mathbf{Z}$ before being fed to the inner product decoder for GAE and VGAE, or being fed to the iterations of Eqs. (8) and (9) for both Graphite-AE and Graphite-VAE.

C. Additional Related Work

Factorization based approaches, such as Laplacian Eigenmaps (Belkin & Niyogi, 2002) and IsoMaps (Saxena et al., 2004), operate on a matrix representation of the graph, such as the adjacency matrix or the graph Laplacian. These approaches are closely related to dimensionality reduction and can be computationally expensive for large graphs.

Random-walk methods are based on variations of the skip-gram objective (Mikolov et al., 2013) and learn representations by linearizing the graph through random walks. These methods, in particular DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover & Leskovec, 2016), learn general-purpose unsupervised representations that have been shown to give excellent performance for semi-supervised node classification and link prediction. Planetoid (Yang et al., 2016) learn representations based on a similar objective specifically for semi-supervised node classification by explicitly accounting for the available label information during learning.