Graph Convolutional Gaussian Processes for Link Prediction

Felix L. Opolka [1] **Pietro Liò** [1]

Abstract

Link prediction aims to reveal missing edges in a graph. We address this task with a deep graph convolutional Gaussian process model. The Gaussian process is transformed using simplified graph convolutions to better leverage the topological information of the graph domain. To scale the Gaussian process model to larger graphs, we introduce a variational inducing point method that places pseudo-inputs on a graph-structured domain. The proposed model represents the first Gaussian process for link prediction that can make use of both node features and topological information. We evaluate our model on three graph data sets with up to thousands of nodes and report consistent improvements over existing Gaussian process models and state-of-the-art graph neural network approaches.

1. Introduction and Related Work

A large variety of real-world scenarios can be modelled by signals that live on the nodes of a graph, from biological networks to communication and social networks (Sen et al., 2008; Kersting et al., 2016). The connective structure of these graphs is not necessarily complete, hence a common task for statistical inference is to infer missing links between nodes.

Recent work in this area (Kipf & Welling, 2016; Zhang & Chen, 2018; Bojchevski & Günnemann, 2018) has focused on methods with two key properties. First, these methods can predict missing links based on both the graph structure itself and a signal that lives on the nodes of the graph, often referred to as the node features. Second, these methods compute node embeddings not only from isolated features of each node but also take into account features in the local neighbourhood of each node, thus providing more context information for predicting missing links.

[1]Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom. Correspondence to: Felix Opolka <flo23@cam.ac.uk>.

ICML Workshop on Graph Representation Learning and Beyond, 2020.

At the core of these methods are usually neural networks equipped with parameterised graph convolution operations. These methods work well when large amounts of labelled data are available, but tend to overfit if labelled data is scarce. In this paper, we present a non-parametric model for link prediction based on deep Gaussian processes, which are less prone to overfitting and, in particular, do not require a validation set for hyperparameter tuning and early stopping.

To increase the inductive bias of the model, the Gaussian process requires adaption to the graph domain. Earlier work by Ng et al. (2018) and Walker & Glocker (2019) introduces Gaussian processes for the graph domain, but focus on node-level or graph-level predictions. Moreover, the kernel proposed by Ng et al. (2018) is limited to one-hop neighbourhoods. To the best of our knowledge, the only existing work on link prediction with Gaussian processes is by Yu & Chu (2008), however it does not consider information from node neighbourhoods.

2. Background

2.1. Single-layer Gaussian Processes

A Gaussian process models functions as samples from an infinite dimensional multivariate normal distribution. The shape of these functions is determined by the mean and covariance (or kernel) function of the process. When modelling observed data $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ with input data matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$, $\mathbf{x}_i \in \mathcal{X}$ and labels $\mathbf{y} \in \mathbb{R}^N$ via Bayesian inference, we can use a Gaussian process as the prior distribution over the latent function:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k_\theta(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $m : \mathcal{X} \to \mathbb{R}$ and $k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ denote the mean and covariance function respectively. The covariance function k_θ is commonly parameterised by a set of hyperparameters θ.

When combined with a Gaussian likelihood $p(y_n|\mathbf{x}_n)$ for each observation $n = 1, \dots, N$, the posterior $p(\mathbf{f}|\mathbf{y}, \mathbf{X})$ is also Gaussian. Predictions for new data points can then be made in a fully Bayesian fashion by marginalising out the latent function $f(\mathbf{x})$. Furthermore, the marginal likelihood $p(\mathbf{y})$ has a closed form solution and can thus be used to optimise the kernel hyperparameters θ, usually via gradient-

based optimisation. For our purposes, we set $\mathcal{X} = \mathbb{R}^D$, hence $\mathbf{X} \in \mathbb{R}^{N \times D}$.

2.2. Deep Gaussian Processes

The expressiveness of a single-layer Gaussian process is constrained by its only kernel function. To overcome this limitation, Damianou & Lawrence (2013) have proposed a deep Gaussian process model, which defines a hierarchy of L layers, where each layer consists of a pre-specified number of Gaussian processes, reminiscent of the units in a neural network layer. The outputs of processes in one layer act as inputs to the processes of the next layer. The last layer consists of as many Gaussian processes as there are outputs. Pseudo-inputs and inducing points (Hensman et al., 2013) are used in each layer to scale the model to large data sets.

The joint density of outputs $\mathbf{Y} \in \mathbb{R}^{N \times O}$, the intermediate latent function values $\mathbf{F}^l$, and the inducing points $\mathbf{U}^l$ is

$$p(\mathbf{Y}, \{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L) = \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{f}_i^L) \prod_{l=1}^L p(\mathbf{F}^l|\mathbf{U}^l, \mathbf{F}^{l-1}, \mathbf{Z}^{l-1}) p(\mathbf{U}^l, \mathbf{Z}^{l-1}) \quad (2)$$

with pseudo-inputs $\{\mathbf{Z}^l\}_{l=1}^L$ and $\mathbf{F}^0 = \mathbf{X}$. Noise between layers is absorbed into the kernel. Due to the non-linearity of the functions modelled by intermediate Gaussian processes, both the posterior and the marginal likelihood are intractable and require a variational approximation.

Salimbeni & Deisenroth (2017) describe a flexible variational approximation of the posterior of a deep Gaussian process, which will be used in this work. The variational posterior is chosen to be

$$q(\{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L) = \prod_{l=1}^L p(\mathbf{F}^l|\mathbf{U}^l; \mathbf{F}^{l-1}, \mathbf{Z}^{l-1}) q(\mathbf{U}^l) \quad (3)$$

with the Gaussian variational distribution $q(\mathbf{U}^l) = \mathcal{N}(\mathbf{U}^l|\mathbf{m}^l, \mathbf{S}^l)$ evaluated at the pseudo-inputs of each layer. The inducing points can be marginalised to give the normal density $q(\{\mathbf{F}^l\}_{l=1}^L)$.

As for a single-layer Gaussian process, we can derive an evidence lower bound (ELBO) to jointly optimise the variational parameters $\{\mathbf{Z}^l, \mathbf{m}^l, \mathbf{S}^l\}_{l=1}^L$ and the kernel hyperparameters θ:

$$\mathcal{L}(\theta, \{\mathbf{Z}^l, \mathbf{m}^l, \mathbf{S}^l\}_{l=1}^L) = \sum_{i=1}^N \mathbb{E}_{q(\mathbf{f}_i^L)} \left[\log p(\mathbf{y}_i|\mathbf{f}_i^L)\right] - \sum_{l=1}^L \mathrm{KL}[q(\mathbf{U}^l) \parallel \mathbf{U}^l; \mathbf{Z}^{l-1}]. \quad (4)$$

The first expectation cannot be evaluated analytically, again because of the non-linear dependency between layers. However, the expectation can be approximated via Monte Carlo sampling by drawing samples from $q(\mathbf{f}_i^L)$, which can be implemented efficiently by drawing samples from a standard normal distribution and applying the reparameterisation trick (Rezende et al., 2014; Kingma et al., 2015) with the mean and variance of $q(\{\mathbf{F}^l\}_{l=1}^L)$ in each layer, starting from the first and propagating the samples through the hierarchy. Due to its form, the ELBO lends itself well to maximisation using stochastic optimisation.

3. Model Description

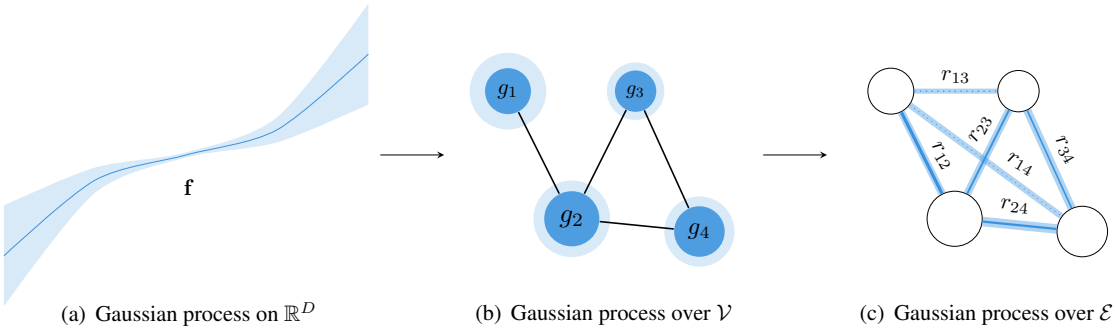
We introduce the deep graph convolutional Gaussian process for link prediction (Link-DGP) in multiple steps. First, we describe a novel single-layer Gaussian process for predictions over nodes (Node-GP). In a second step, building on Node-GP, we introduce a single-layer Gaussian process for predictions over pairs of nodes (Link-GP), along with an effective inducing point method. Finally, we use these two building blocks as Gaussian process units in the layers of the deep Gaussian process model.

3.1. Gaussian process over nodes

We aim to define a Gaussian process kernel that is capable of seizing the inductive bias of the domain whose structure is given by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of vertices $\mathcal{V}$, $|\mathcal{V}| = N$, and a set of edges $\mathcal{E}$, $|\mathcal{E}| = E$. The graph structure is further described by the adjacency matrix $\mathbf{A}$ without self-loops, i.e. its diagonal entries are 0. Input data is given in form of a signal $\mathbf{X} \in \mathbb{R}^{N \times D}$ living on said domain.

In a non-probabilistic setting, adaption to the graph domain is commonly achieved by convolving the node features using graph convolutions, as proposed by Kipf & Welling (2017). Wu et al. (2019) propose a simplified form of the graph convolution $\mathbf{g} = \tilde{\mathbf{S}}^K \mathbf{X}\mathbf{w}$ with weights $\mathbf{w} \in \mathbb{R}^{D \times 1}$ and *convolution matrix* $\tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$, which is raised to the K^{th} matrix power. Here, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. The convolution matrix $\tilde{\mathbf{S}}$ acts as a smoothing device on the input features, thus biasing the hidden representations to vary less within a neighbourhood. This increases the inductive bias of the model under the assumption that labels within a node neighbourhood are more likely to be similar.

Building on this idea, we obtain the corresponding probabilistic formulation by placing a multivariate Gaussian prior on the weights $\mathbf{w}$. Furthermore, we can transform the input signal using a feature map $\phi_\theta : \mathbb{R} \to \mathcal{H}$ that maps inputs to a potentially infinite-dimensional Hilbert space $\mathcal{H}$ and is parameterised by a set of hyperparameters θ. By subsequently marginalising the weights $\mathbf{w}$, we obtain an equivalent formulation $\mathbf{g} = \tilde{\mathbf{S}}^K \mathbf{f}$, where $\mathbf{f} \in \mathbb{R}^{N \times 1}$ is normally distributed with covariance matrix $[\mathbf{K}]_{ij} = \langle \phi_\theta(\mathbf{x}_i), \phi_\theta(\mathbf{x}_j) \rangle_{\mathcal{H}}$ and we



(a) Gaussian process on $\mathbb{R}^D$ (b) Gaussian process over $\mathcal{V}$ (c) Gaussian process over $\mathcal{E}$

Figure 1. Overview of the proposed graph convolutional Gaussian processes over nodes (b) and edges (c). We start with a regular Gaussian process $\mathbf{f}$ (a) operating solely on the node features that is oblivious to the graph structure. Each node feature is treated as an observation on the Euclidean domain $\mathbb{R}^D$. This Gaussian process is transformed using simplified graph convolutions to yield a graph convolutional Gaussian process $\mathbf{g}$ over the nodes $\mathcal{V}$ of the graph (b). Finally, a series of such graph convolutional Gaussian processes yields a graph convolutional Gaussian process $\mathbf{r}$ over edges (c). Function values in (b) and (c) are expressed through the size of the nodes and the thickness of the links respectively. Confidence intervals are sketched in light blue. A hierarchy of Gaussian processes from (b) and (c) can be combined to form a deep Gaussian process for link prediction.

assume $\mathbf{f}$ has zero mean. The simplified graph convolution acts as a linear transformation on $\mathbf{f}$, hence the distribution of the resulting signal $\mathbf{g}$ is also Gaussian:

$$\mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, (\tilde{\mathbf{S}}^K)\mathbf{K}(\tilde{\mathbf{S}}^K)^T\right). \tag{5}$$

Thus, $\mathbf{g}$ corresponds to a Gaussian process on the domain whose structure is given by the graph $\mathcal{G}$. The covariance matrix $\mathbf{K}$ is computed by the *node feature kernel* $k_\theta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. When fixing $K = 1$ and using an asymmetric normalisation for the convolution matrix $\tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, we recover the Gaussian process model for semi-supervised node classification described by Ng et al. (2018) as a special case.

Counteracting oversmoothing While a certain degree of smoothing is desirable to increase the inductive bias of the model, too much smoothing inhibits the ability of the model to describe more complex relationships between nodes in a neighbourhood, thus degrading its overall performance. This phenomenon is commonly referred to as *oversmoothing* and has been the subject of several recent studies (Li et al., 2018; Wu et al., 2019; Xu et al., 2018).

We propose to counter oversmoothing by taking advantage of the ability of the Gaussian process to optimise hyperparameters to select between the number of graph convolutions to be applied. We achieve this by smoothly interpolating in each convolution step between the convolution matrix $\tilde{\mathbf{S}}$ and the identity matrix. The k^{th} convolution matrix hence becomes $\tilde{\mathbf{S}}_k = \lambda_k\tilde{\mathbf{S}} + (1 - \lambda_k)\mathbf{I}$, where $\lambda = [\lambda_1, \ldots, \lambda_K] \in [0; 1]^K$ are hyperparameters, subsequently referred to as the *convolution weights*. The final Gaussian process prior thus becomes

$$\mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, (\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}(\tilde{\mathbf{S}}_1^T \cdots \tilde{\mathbf{S}}_K^T)\right). \tag{6}$$

A visualisation of the graph convolutional Gaussian process over nodes is shown in Figure 1 (b).

3.2. Gaussian process over pairs of nodes

A Gaussian process model over edges of an undirected graph must operate on the domain of pairs of nodes such that it is invariant to the order of the nodes within the pair. Yu & Chu (2008) propose to model edges using a Gaussian process

$$r(\mathbf{x}_i, \mathbf{x}_j) \sim \mathcal{GP}(\mathbf{0}, c((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}'_i, \mathbf{x}'_j))) \tag{7}$$

with kernel $c((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}'_i, \mathbf{x}'_j)) = k(\mathbf{x}_i, \mathbf{x}'_i)k(\mathbf{x}_j, \mathbf{x}'_j) + k(\mathbf{x}_i, \mathbf{x}'_j)k(\mathbf{x}_j, \mathbf{x}'_i)$, which is the result of a series of Gaussian processes over nodes with kernel function $k(\cdot, \cdot)$.

As we would like the link prediction Gaussian process to incorporate neighbourhood information in its predictions, we use the graph convolutional kernel from Equation 6 as the node kernel function $k(\cdot, \cdot)$. This results in the single-layer graph convolutional Gaussian process r for link prediction:

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \mathbf{K}) \tag{8}$$

$$\mathbf{g}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \hat{\mathbf{K}} \equiv (\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}(\tilde{\mathbf{S}}_1^T \cdots \tilde{\mathbf{S}}_K^T)) \tag{9}$$

$$\mathbf{r}(\mathbf{x}_i, \mathbf{x}_j) \sim \mathcal{GP}(\mathbf{0}, \mathbf{C}),$$
$$\text{with } \mathbf{C}_{(i,j)(i',j')} = \hat{\mathbf{K}}_{ii'}\hat{\mathbf{K}}_{jj'} + \hat{\mathbf{K}}_{ij'}\hat{\mathbf{K}}_{ji'}. \tag{10}$$

As before, $\mathbf{K}$ is computed using the node feature kernel $k_\theta : \mathbb{R}^{D\times 1} \times \mathbb{R}^{D\times 1} \to \mathbb{R}$ on the input node features. The full Link-GP model is visualised in Figure 1.

Method	USAir		C.ele		Router	
	AUC	AP	AUC	AP	AUC	AP
VGAE	89.24	91.46	83.76	81.94	76.79	83.88
graph2gauss	88.79	89.36	85.59	83.10	75.12	81.31
Link-GP (ours)	95.39	89.64	90.47	75.37	89.22	84.01
Link-DGP-2 (ours)	98.03	91.32	93.78	81.85	90.61	91.38
Link-DGP-3 (ours)	97.96	**93.46**	**94.82**	83.82	90.73	91.92
Link-DGP-4 (ours)	**98.16**	92.24	93.90	**85.34**	**94.16**	**92.46**

Table 1. Experimental results for the proposed graph-convolutional Gaussian process for link prediction in its single-layer version (Link-GP) and in its multi-layer version (Link-DGP), with varying number of layers. We compare the results to the variational graph auto-encoder (Kipf & Welling, 2016) and the graph2gauss model (Bojchevski & Günnemann, 2018), in terms of area under the ROC curve (**AUC**) and average precision (**AP**).

Variational inducing point approximation When using our model either as a standalone Gaussian process or as part of a deep Gaussian process, we wish to leverage inducing points to address an intractable posterior and scale the model to large data sets.

However, naively placing a set of of M pseudo-inputs onto the signal domain $\mathbb{R}^{D\times 1}$ fails because of the functional form of the kernel c, which expects separate inputs for the two nodes of an edge. Hence, we require *inducing edges* that are represented by pairs of inducing points. We solve this problem by constructing a randomly generated, connected *inducing graph* $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ with $|\bar{\mathcal{V}}| = \bar{N}$ and $|\bar{\mathcal{E}}| = \bar{E}$ and placing pseudo-input $\mathbf{z}_i \in \mathbb{R}^{D\times 1}$ on each of the $\bar{N}$ nodes in the inducing graph. Each pseudo-input represents a node feature on the inducing graph. These pseudo-node features are optimised such that the inducing edges are most informative for posterior inference of missing links in the input graph.

The pseudo-inputs $\mathbf{z}_i \in \mathbb{R}^{D\times 1}$ on the nodes of the inducing graph are placed onto the domain of $\mathbf{f}$. As the real input points, unlike the pseudo-inputs, are subject to a graph convolution, we have to employ straightforward inter-domain inference (Lázaro-Gredilla & Figueiras-Vidal, 2009; van der Wilk et al., 2017) for predicting missing links.

3.3. Deep Graph Convolutional Gaussian Process for Link Prediction

The model described by Equation 10 can readily be used for link prediction tasks. However, higher model expressiveness can be achieved by deep Gaussian process models, thus more accurately capturing the mapping from node features to linkage information. The two Gaussian process models introduced so far, graph convolutional Gaussian process over nodes (Node-GP) from Equation 6 and graph convolutional Gaussian process over links (Link-GP) from Equation 10, will be used as units within the deep Gaussian process layers. For a deep Gaussian process with a total of L layers, we propose to use Node-GP units in the first $L-1$ layers followed by a final layer consisting of a single Link-GP unit. The first $L-1$ layers are responsible for extracting node representations and the final layer derives from these a distribution over a pair of nodes.

4. Results and Conclusion

We apply our method to a set of benchmark data sets for link prediction covering a wide range of domains. The data sets are USAir (Batagelj & Mrvar, 2006), C.ele (Watts & Strogatz, 1998), and Router (Spring et al., 2004). An overview of the data set statistics along with details on the training setup are provided in the appendix. Notably, Gaussian processes tend not to suffer from overfitting, hence it is not necessary to set aside a validation data set. We compare the results of our single-layer and deep graph convolutional Gaussian processes to those of the variational graph auto-encoder (Kipf & Welling, 2016) and the graph2gauss model (Bojchevski & Günnemann, 2018) in Table 1.

We find that the single-layer graph convolutional Gaussian process (Link-GP) outperforms the baseline models on all three data sets in terms of AUC but falls short in terms of AP on the USAir and C.ele data set. The deep graph convolutional Gaussian process outperforms all baselines both in terms of AUC and AP, often by a large margin. We also find that the deep Gaussian process (Link-DGP) achieves better results compared to its single-layer variant on all data sets and both in terms of AUC and AP. Performance differences between deep Gaussian processes with varying number of layers are small, with the best results achieved with three to four layers.

The results show that the deep Gaussian processes clearly outperform the parametric baselines. We also note performance benefits of using a hierarchy of Gaussian processes over a single-layer Gaussian process. Deeper Gaussian processes with more than two layers tend to perform better, however the differences are less pronounced. We investigate the effect of deeper hierarchies, among other model properties, in more detail in an upcoming full-length paper.

References

Batagelj, V. and Mrvar, A., 2006. URL http://vlado.fmf.uni-lj.si/pub/networks/data/.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Damianou, A. and Lawrence, N. Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 2013.

Erdös, P. and Rényi, A. On random graphs I. *Publicationes Mathematicae Debrecen*, 1959.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, 2017.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2013.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. Benchmark data sets for graph kernels, 2016. URL http://graphkernels.cs.tu-dortmund.de.

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

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*, 2015.

Kipf, T. N. and Welling, M. Variational graph auto-encoders, 2016. arXiv preprint arXiv: 1611.07308.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. Inter-domain gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems 22*, 2009.

Li, Q., Han, Z., and ming Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Ng, Y. C., Colombo, N., and Silva, R. Bayesian semi-supervised learning with graph gaussian processes. In *Advances in Neural Information Processing Systems 31*, 2018.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Salimbeni, H. and Deisenroth, M. Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems 30*, 2017.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 2008.

Spring, N., Mahajan, R., Wetherall, D., and Anderson, T. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 2004.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems 30*, 2017.

Walker, I. and Glocker, B. Graph convolutional Gaussian processes. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world' networks. *Nature*, 1998.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Yu, K. and Chu, W. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems 20*, 2008.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems 31*, 2018.

Data	# nodes	# edges	average node degree
USAir	332	2,126	12.81
C.ele	297	2,148	14.46
Router	5,022	6,258	2.49

Table 2. Statistics of the data sets used in our experiments.

A. Training Setup

The data set statistics are shown in Table 2. The training setup for the models compared in Section 4 are as follows:

Baselines For the variational graph auto-encoder and the graph2gauss model, we use the implementation provided by the authors with the hyperparameters described in the paper. Both can be used without a validation set, which is the option we choose for our evaluation.

Single-layer Gaussian processes In all our experiments, we set the maximum number of convolutions to $K = 2$. A random, connected inducing graph with $|\bar{V}| = \frac{|V|}{8}$ nodes and $|\bar{E}| = 2|\bar{V}|$ edges is drawn from an Erdős-Rényi model (Erdös & Rényi, 1959). The pseudo-inputs are initialised with K-means on the training data set. We use a constant mean function and a polynomial kernel of degree three for the base kernel of the graph convolutional Gaussian process. The convolution weights λ_1 and λ_2 are initialised to 0.5 and 0.3 respectively. As the data sets come without node features, we use `node2vec` embeddings of size 128, following (Zhang & Chen, 2018). To improve scalability, we sample nodes from the neighbourhoods of the nodes incident to the target edge, which will then form the domain of the convolution operation (Hamilton et al., 2017). We sample up to 20 distinct 1-hop neighbours and up to 10 distinct 2-hop neighbours. For parameter optimisation, we use the Adam optimiser (Kingma & Ba, 2015) with a learning rate of 0.005. We train our models for up to 1000 epochs with a batch size of 256 edges and stop training early if there is no improvement in ELBO over 100 epochs. All experiments were performed on a NVIDIA Titan X GPU with Pascal architecture and 12GB of memory.

Deep Gaussian processes For hyperparameters that the single-layer Link-GP and the deep Link-DGP have in common, we use the same hyperparameter values, with the exception of the base kernel of the graph convolutional Gaussian processes, where we use a radial basis function kernel (RBF) with automatic relevance determination (ARD)

$$k(\mathbf{x}, \mathbf{x}') = \nu \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{l_d^2}\right). \qquad (11)$$

Its variance ν and lengthscales l_d are initialised to 1.0. Each hidden layer of the Link-DGP consists of 32 Gaussian process units. A single sample is used to approximate the likelihood term of the ELBO (cf. Equation 4) during training and 50 during testing. Further model details follow the approach by (Salimbeni & Deisenroth, 2017). The mean functions of hidden layers are initialised to linear maps $m(\mathbf{X}) = \mathbf{X}\mathbf{W}$. If the input dimensions of the layer is less or equal to the output dimension, the (potentially zero-padded) identity matrix is used for $\mathbf{W}$. If the input dimensions is greater than the output dimension, $\mathbf{W}$ is the PCA mapping with as many eigenvectors as the output dimensionality of the layer. Inducing means $\mathbf{m}^l$ are initialised to $\mathbf{0}$ and inducing variances $\mathbf{S}^l$ to the identity matrix, scaled by 10^{-5} for hidden layers.

Deep Gaussian processes often require training with a cold posterior for the first few epochs, referring to training with a scaled down KL-term in Equation 4. We initially optimise the ELBO without the KL-term until the training likelihood has decreased by 10% compared to its value after the first epoch and then increase the KL-scaling term from 0.0 to 1.0 over 100 epochs using x^5 for interpolation.