Bayesian Link Prediction with Deep Graph Convolutional Gaussian Processes

Felix L. Opolka
University of Cambridge
& Invenia Labs

Pietro Liò
University of Cambridge

Abstract

Link prediction aims to reveal missing edges in a graph. We introduce a deep graph convolutional Gaussian process model for this task, which addresses recent challenges in graph machine learning with oversmoothing and overfitting. Using simplified graph convolutions, we transform a Gaussian process to 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. Multiple Gaussian processes are assembled into a hierarchy whose structure allows skipping convolutions and thus counteracting oversmoothing. The proposed model represents the first Gaussian process for link prediction that makes use of both node features and topological information. We evaluate our model on multiple graph data sets with up to thousands of nodes and report consistent improvements over competitive link prediction approaches.

1 INTRODUCTION

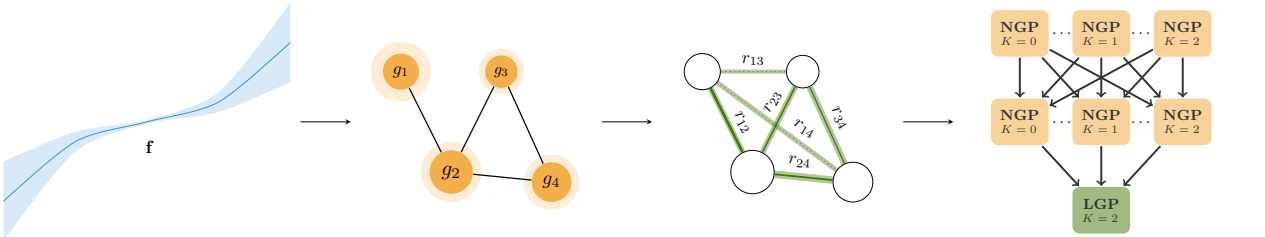
A large variety of real-world scenarios can be modelled by signals that live on the nodes of a graph: from biological networks to communication and social networks (Sen et al., 2008; Kersting et al., 2016). The connective structure of these graphs is not necessarily complete, hence a common task for statistical inference is to infer missing links between nodes (Wang et al., 2015). In a protein-protein interaction network, for example, link prediction is used to suggest interactions between proteins in a cell (Lei and Ruan, 2012).

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

Recent work in this area (Kipf and Welling, 2016; Zhang and Chen, 2018; Bojchevski and Günnemann, 2018) has focused on methods with two key properties. Firstly, these methods can predict missing links based on both the graph structure itself and a signal in the nodes of the graph, often referred to as the node features. Secondly, 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. At the core of these methods are usually neural networks equipped with parameterised graph convolution operations (Defferrard et al., 2016; Kipf and Welling, 2017).

A number of recent works have investigated limitations of neural networks fitted with graph convolutions (Li et al., 2018; Wu et al., 2019b; Xu et al., 2018; Klicpera et al., 2019; Rong et al., 2020). In particular, Rong et al. (2020) have identified two key challenges for graph convolutional networks. Firstly, graph convolutional networks tend to *overfit* on smaller graphs or graphs with few labelled nodes and edges, causing bad generalisation performance. Secondly, while convolutions are useful because they smooth the graph signal, meaning nodes closer together in the graph take on more similar values (e.g. hidden representations), applying too many convolutions reduces the expressivity of the network, making it unable to model variations within a node neighbourhood; a problem referred to as *oversmoothing*. The appropriate degree of smoothing depends on the data set, however graph convolutional networks have a rigid structure with a fixed number of convolutions and are therefore unable to adaptively learn the most suitable degree of smoothing.

We address both challenges with a new Bayesian graph convolutional model building on non-parametric Gaussian processes fitted by maximising the marginal likelihood, thus making it more robust against overfitting. In fact, our model does not require a validation set for training. We achieve high expressivity by extending the model to a deep Gaussian process constructed in a



(a) Gaussian process on $\mathbb{R}^D$ (b) Gaussian process over $\mathcal{V}$ (Node-GP/ NGP) (c) Gaussian process over $\mathcal{V} \times \mathcal{V}$ (Link-GP/ LGP) (d) Deep Gaussian process over $\mathcal{V} \times \mathcal{V}$ (Link-DGP)

Figure 1: Visualisation of the derivation of the deep graph convolutional Gaussian process model. 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 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 lighter colours. A hierarchy of Gaussian processes from (b) and (c) can be combined to form a deep Gaussian process for link prediction (d). $K = 0, 1, 2$ indicates the number of convolutions this particular Node-GP or Link-GP applies to the incoming signal.

way such that the size of the pre-domain of the convolution operation is a property orthogonal to the number of layers of the deep Gaussian process model. This allows independently adjusting the degree of smoothing and the model expressivity. We further introduce hyperparameters that control the degree of smoothing, which are learned directly from the data set using type II maximum likelihood maximisation. Recent results suggest link prediction tasks can benefit particularly from modelling multi-hop neighbourhoods (Yue et al., 2019), making them particularly prone to oversmoothing. Link prediction is therefore the main focus of this work. Furthermore, many link prediction applications can naturally make use of the confidence estimates output by our Gaussian process model. In the aforementioned example of protein-protein interaction prediction, the predictive uncertainty can be used to prioritise which interactions to practically confirm in a wet-lab experiment.

We derive the deep graph convolutional Gaussian process for link prediction in three steps. In Section 4.1, we define a graph convolutional Gaussian process model over the *nodes* of a graph, referred to as *Node-GP*, by transforming a Gaussian process defined on the Euclidean domain with graph convolutions. It can be trained to automatically fit the neighbourhood size of the graph convolutions to the input graph. In Section 4.2, we further adapt the resulting model over nodes to a Gaussian process over *pairs of nodes*, thus suitable for link prediction. We refer to this model as *Link-GP*. We suggest a variational inducing point method for link prediction that works by placing inducing points on the nodes of an inducing graph. Finally, in Section 4.3, we assemble multiple instances of Node- and Link-GPs into a deep hierarchy. Each layer performs convolutions with different neighbourhood sizes, effectively de-coupling model expressivity and the degree of smoothing and thus preventing oversmoothing. In Section 5, we evaluate our model on six benchmark data sets with up to thousands of nodes and edges, often strongly outperforming neural network-based models. We also assess the quality of the uncertainty estimates as well as the data-efficiency of the model.

2 Background

2.1 Single-layer Gaussian Processes

A Gaussian process (Rasmussen and Williams, 2005) models functions as samples from an infinite dimensional multivariate normal distribution. The shape of the functions are 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 $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, $\mathbf{x}_i \in \mathcal{X}$ 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 $p(y_n|\mathbf{f}_n)$ for each observation $n = 1, \ldots, N$, 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 $\mathbf{X} \in \mathbb{R}^{N \times D}$.

This formulation of Gaussian processes is limited in two ways. Firstly, when the likelihood is not Gaussian, as is the case for link prediction, neither the posterior distribution nor the marginal likelihood have a closed-form solution. Secondly, inference with a Gaussian process requires the inversion of an $N \times N$ matrix, which has complexity $\mathcal{O}(N^3)$ and is thus infeasible for large data sets. Following Hensman et al. (2015), both problems are commonly addressed by approximating the intractable posterior with a variational posterior distribution evaluated at a small set of *pseudo-inputs* $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_M]^\top$, with $\mathbf{z}_i \in \mathcal{X}$ and $M \ll N$. The function values $\mathbf{u} = [f(\mathbf{z}_1), \dots, f(\mathbf{z}_M)]^\top$ at these inputs are referred to as *inducing points* and are assumed to follow the same Gaussian process prior distribution as the original outputs, which means that $p(\mathbf{u}) = \mathcal{N}(\mathbf{m}_\mathbf{z}, \mathbf{K}_{\mathbf{zz}})$, where $[\mathbf{m}_\mathbf{z}]_i = m(\mathbf{z}_i)$ and $[\mathbf{K}_{\mathbf{zz}}]_{ij} = k_\theta(\mathbf{z}_i, \mathbf{z}_j)$. Inference for a new input $\mathbf{x}^*$ is now performed using the *sparse* Gaussian process over the inducing points:

$$f(\mathbf{x}^*)|\mathbf{u} \sim \mathcal{GP}(\mathbf{k}_{\mathbf{zx}^*}^\top \mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{u}, k_\theta(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_{\mathbf{zx}^*}^\top \mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{k}_{\mathbf{zx}^*}),$$

where $\mathbf{k}_{\mathbf{zx}^*} = [k_\theta(\mathbf{z}_1, \mathbf{x}^*), \dots, k_\theta(\mathbf{z}_M, \mathbf{x}^*)]$ and we use $m(\mathbf{x}) = 0$.

The variational distribution is chosen to be a multivariate Gaussian distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$. The pseudo-inputs $\mathbf{Z}$, as well as $\mathbf{m}$ and $\mathbf{S}$ are variational parameters, which are optimised jointly with the kernel hyperparameters θ by maximizing the Evidence Lower Bound (ELBO) objective:

$$\begin{aligned}\mathcal{L}(\theta, \mathbf{Z}, \mathbf{m}, \mathbf{S}) = &\sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n|f(\mathbf{x}_n))] \\ &- \mathrm{KL}[q(\mathbf{u}) \parallel p(\mathbf{u})]. \end{aligned} \tag{2}$$

The shape of this objective enables optimisation via stochastic gradient descent, which reduces the memory complexity of an individual update step, thus allowing us to train on larger data sets.

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 and Lawrence (2013) proposed a deep Gaussian process model, which defines a hierarchy of L layers, where each layer consists of a prespecified 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 final 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

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

with pseudo-inputs $\{\mathbf{Z}^l\}_{l=1}^L$ and $\mathbf{F}^0 = \mathbf{X}$. Noise between layers is absorbed into the kernel.

2.3 Graph Convolutions

Graph convolutions are the result of an effort to generalise the parameterised convolution operations for images (LeCun et al., 1999) to the domain of general graphs (Bruna et al., 2014). The graph convolution is applied to an input signal $\mathbf{x} \in \mathbb{R}^N$ lying on the domain of a graph $\mathcal{G}$ with N nodes and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ (without self-loops). The convolution operator is formulated as a multiplication of the filter with the input signal mapped to the spectral domain via the Fourier transform. Analogously to the Fourier transform on the Euclidean domain, the graph Fourier transform is defined as the decomposition of a signal into the eigenfunctions of the Laplace operator. On the graph domain, this operator is given by the Laplace matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$, where $\mathbf{D} \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix with $D_{ii} = \sum_{j=1}^{N} A_{ij}$. To localise the convolution operation, the filter is commonly parameterised with polynomials in the spectral domain (Defferrard et al., 2016). We will use a variant of the graph convolution operation to incorporate topological information into the Gaussian process model.

3 RELATED WORK

Our work is closely related to the two fields of Gaussian processes for graph-structured data and link prediction.

Gaussian processes for graph-structured data have previously been studied under the term relational learning. These methods have been applied to semi-supervised classification of nodes in a graph, such as the relational Gaussian process (Chu et al., 2007), the mixed graph Gaussian process (Silva et al., 2008), or the label propagation algorithm (Zhu et al., 2003a,b). Inspired by graph neural networks, more recent work

has developed Gaussian process models that explicitly consider nodes together with node features in their local neighbourhood. The graph Gaussian process described by Ng et al. (2018) computes node representations by averaging the node features of the 1-hop neighbourhood and subsequently performing semi-supervised node classification. Unlike the graph convolutional Gaussian process proposed here, it only considers 1-hop node neighbourhoods, thus limiting the node neighbourhood information accessible to the model. Recently, Gaussian processes have also been used for predicting multi-output signals on graphs (Zhi et al., 2020; Li et al., 2020). The graph convolutional Gaussian process introduced by Walker and Glocker (2019) employs graph convolutions to produce representations of patches in the graph and sums up these patches via an additive Gaussian process model. Unlike the models described so far, it is used for graph-level prediction such as image or mesh classification. Existing Gaussian process models for link prediction either do not include information from node neighbours (Yu and Chu, 2008) or do not consider node attributes (Lloyd et al., 2012), which may restrict their predictive performance.

Link prediction models A common class of link prediction methods are heuristic-based models, which compute a heuristic for node similarity and output it as the likelihood of a link, as explored systematically by Zhang and Chen (2018). Other methods focus on predicting links based on latent node features that are derived from the graph structure. These include matrix factorisation (MF) (Koren et al., 2009), the stochastic block model (SBM) for link prediction (Airoldi et al., 2008b), and the spring-electrical model for link prediction (Kashinskaya et al., 2019). More recent approaches such as DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016) rely on random walks to produce node embeddings that encode latent features and pair-wise comparison of the embeddings to predict links. These approaches can also be cast as matrix factorisation (Qiu et al., 2018). Naturally, matrix factorisation methods do not consider node features.

Another class of link prediction methods makes use of neural networks. The Weisfeiler-Lehman Neural Machine (WLNM) (Zhang and Chen, 2017) trains a fully-connected neural network on adjacency matrices. SEAL (Zhang and Chen, 2018) employs graph-neural networks in a non-probabilistic setting. The network operates on node features and hand-crafted node labels that indicate a node's role in its neighbourhood. Most similar to our model, the graph variational auto-encoder by Kipf and Welling (2016) combines probabilistic modelling and graph convolutions, thus also considering neighbourhood information. It uses a graph neural network (Kipf and Welling, 2017) as the encoder and a dot product decoder. Newer variants have been suggested more recently (Davidson et al., 2018; Hasanzadeh et al., 2019). The graph2gauss model by Bojchevski and Günnemann (2018) embeds nodes as Gaussian distributions output by a graph convolutional network. The embedding training ensures that the distance between embeddings reflects the geodesic distance of nodes on the graph.

The Gaussian process model proposed in the following sections exhibits many of the individual strong points of existing models. It considers both graph structure and node features and incorporates local neighbourhood information when inferring missing links. Moreover, the Bayesian inference framework provides us with a principled way of obtaining uncertainty estimates for our predictions.

4 METHODOLOGY

We introduce the deep graph convolutional Gaussian process for link prediction (Link-DGP) in three steps. (1) We describe a single-layer Gaussian process for predictions over nodes (Node-GP). (2) 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. (3) We use these two building blocks as Gaussian process units in the layers of the deep Gaussian process model.

4.1 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 equivalently 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, adaptation to the graph domain is commonly achieved by convolving the node features using graph convolutions, as suggested by Kipf and Welling (2017). Wu et al. (2019a) propose a simplified form of the graph convolution

$$\mathbf{g} = \tilde{\mathbf{S}}^K \mathbf{X} \mathbf{w} \tag{3}$$

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 $\tilde{\mathbf{A}}$. Intuitively, multiplying with the convolution matrix computes hidden representations that are the weighted average of the features of a node and the features of its immediate

neighbours. Consequently, applying the convolution matrix K times results in hidden representations computed from the K-hop neighbourhood.

The probabilistic equivalent of the convolution model in Equation 3 can be obtained by placing a multivariate Gaussian prior on the weights $\mathbf{w}$. Furthermore, we can transform the input signal using a feature map $\phi_\theta : \mathbb{R}^D \to \mathcal{H}$ 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 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:

$$\mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, (\tilde{\mathbf{S}}^K)\mathbf{K}(\tilde{\mathbf{S}}^K)^\top\right). \tag{4}$$

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* $k_\theta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$.

Wu et al. (2019a) have demonstrated that applying the convolution matrix $\tilde{\mathbf{S}}$ to a non-probabilistic linear model acts as a smoothing device on the input features, thus biasing the hidden representations to vary less within a neighbourhood. The larger the exponent K of the convolution matrix, the stronger the smoothing effect. We make a similar observation when convolving a stochastic process: higher exponents K lead to smoother functions being sampled from the Gaussian process prior. We study this in more detail in Appendix A. In both cases, applying the convolution improves the inductive bias of the model under the assumption that labels within a node neighbourhood are more likely to be similar.

In practice, different tasks and data sets call for different degrees of smoothing. Whereas for graph neural networks, the number of convolutions has to be specified ahead of training, for the graph convolutional Gaussian process we can take advantage of type II maximum likelihood maximisation 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*, which control to what degree the convolutions should be applied. The final Gaussian process prior thus becomes

$$\mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, (\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}(\tilde{\mathbf{S}}_1^\top \cdots \tilde{\mathbf{S}}_K^\top)\right). \tag{5}$$

We refer to the model as *Node-GP* and visualise it in Figure 1 (b).

4.2 Gaussian process over pairs of nodes

The model described so far defines a Gaussian process over the nodes of the graph $\mathcal{G}$. In the following, we describe how to transform such a Gaussian process over nodes into a Gaussian process over node pairs to predict potential edges between them. We further introduce a variational inducing point approximation for the intractable posterior.

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. Reminiscent of matrix factorisation, functions over node pairs can be modelled by a matrix product

$$r(\mathbf{x}_i, \mathbf{x}_j) = L^{-\frac{1}{2}} \sum_{l=1}^{L} g_l(\mathbf{x}_i)g_l(\mathbf{x}_j) - L^{\frac{1}{2}}k(\mathbf{x}_i, \mathbf{x}_j), \tag{6}$$

where $\{g_l\}_{l=1}^L$ is a set of independent, identically distributed random variables with $g_l(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}'))$ modelling functions over the nodes of the graph (Yu and Chu, 2008). In the limit of $L \to \infty$, r converges to a Gaussian process over node pairs:

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

with kernel $c\left((\mathbf{x}_i, \mathbf{x}_j), (\mathbf{x}'_i, \mathbf{x}'_j)\right) = k(\mathbf{x}_i, \mathbf{x}'_i)k(\mathbf{x}_j, \mathbf{x}'_j) + k(\mathbf{x}_i, \mathbf{x}'_j)k(\mathbf{x}_j, \mathbf{x}'_i)$ according to Yu and Chu (2008, Theorem 2.2). Crucially, the resulting Gaussian process has the desired property that its kernel c is invariant to the order of the nodes within a pair.

As we would like the link prediction Gaussian process to incorporate neighbourhood information in its predictions, we define the set of random variables $\{g_l\}_{l=1}^L$ in Equation 6 to follow the graph convolutional Gaussian process prior defined in Equation 5. This results in the final formulation of the Link-GP model:

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

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

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

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. We visualise the model in Figure 1 (c).

Variational inducing point approximation Predicting potential links between node pairs boils down to a binary classification problem, which dictates a

Bernoulli likelihood. This leads to an intractable posterior distribution, which we will approximate with a variational distribution. We will also use a set of M inducing points to reduce the computational complexity of inference (see Section 4.3 for a detailed discussion). However, naïvely placing a set of of M inducing points onto the signal domain $\mathbb{R}^{D\times 1}$ fails because of the functional form of the kernel c, which is defined on the domain of node pairs. Hence, we require *inducing edges* that are represented by pairs of inducing points. We solve this problem by constructing an *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-inputs $\mathbf{z}_i \in \mathbb{R}^{D\times 1}$ on each of the $\bar{N}$ nodes in the inducing graph. Each pseudo-input represents a node feature on the inducing graph.

In our experiments, we obtain the inducing graph by sampling from an Erdős-Rényi model (Erdös and Rényi, 1959) with $\bar{E}$ edges. We note that the exact topology of the inducing graph is of little importance and we have found that altering the type of random graph model does not affect the predictive performance. Unlike the input data, no graph convolutions are applied to the inducing points, hence the inducing graph merely forms the domain on which the inducing points lie. It is the pseudo-inputs, i.e. the node features of the inducing graph, which 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 inter-domain inference (Lázaro-Gredilla and Figueiras-Vidal, 2009; van der Wilk et al., 2017) for predicting missing links. The inter-domain covariance between a node pair $(\mathbf{x}_i, \mathbf{x}_j)$ of the input graph and a node pair $(\mathbf{z}_{i'}, \mathbf{z}_{j'})$ of the inducing graph is given by

$$\begin{aligned}\mathbf{C}_{(i,j)(i',j')} = &[(\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}_{\mathbf{XZ}}]_{ii'}[(\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}_{\mathbf{XZ}}]_{jj'} \\ &+ [(\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}_{\mathbf{XZ}}]_{ij'}[(\tilde{\mathbf{S}}_1 \cdots \tilde{\mathbf{S}}_K)\mathbf{K}_{\mathbf{XZ}}]_{ji'}. \qquad (11)\end{aligned}$$

$\mathbf{K}_{\mathbf{XZ}}$ is computed using the node feature kernel $k_\theta : \mathbb{R}^{D\times 1} \times \mathbb{R}^{D\times 1} \to \mathbb{R}$ applied to node features of the input graph and the inducing graph.

4.3 Deep graph convolutional Gaussian process

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 and graph topology to linkage information. The two Gaussian process models introduced so far, the graph convolutional Gaussian process over nodes (Node-GP) from Equation 5 and the 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. The hierarchy of the deep Gaussian process is visualised in Figure 1 (d).

The posterior of the resulting deep Gaussian process is intractable due to the highly non-linear kernel functions and we thus resort to an approximation following the variational framework introduced by Salimbeni and Deisenroth (2017). The variational posterior is chosen to be

$$\begin{aligned}q(\mathbf{r}^L, \mathbf{U}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}) = &\, p(\mathbf{r}^L|\mathbf{U}^L; \mathbf{G}^{L-1}, \mathbf{Z}^{L-1}) \\ &\cdot q(\mathbf{U}^L) \prod_{l=1}^{L-1} p(\mathbf{G}^l|\mathbf{U}^l; \mathbf{G}^{l-1}, \mathbf{Z}^{l-1}) q(\mathbf{U}^l) \qquad (12)\end{aligned}$$

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. Crucially, the pseudo-inputs of the final layer are placed on the domain of an inducing graph as described above. The inducing points can be marginalised to give the normal density $q(\mathbf{r}^L, \{\mathbf{G}^l\}_{l=1}^{L-1})$. A more detailed derivation of the variational approximation can be found in Appendix G.

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:

$$\begin{aligned}\mathcal{L}(\theta, \{\mathbf{Z}^l, \mathbf{m}^l, \mathbf{S}^l\}_{l=1}^L) = &\sum_{e_{ij}\in\mathcal{T}} \mathbb{E}_{q(r_{ij}^L)}\left[\log p(y_{ij}|r_{ij}^L)\right] \\ &- \sum_{l=1}^{L-1} \mathrm{KL}[q(\mathbf{U}^l) \parallel p(\mathbf{U}^l; \mathbf{Z}^{l-1})]. \qquad (13)\end{aligned}$$

Here, $\mathcal{T}$ is the training set of node pairs and $p(y_{ij}|r_{ij}^L)$ is a Bernoulli likelihood. 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 $q(\mathbf{g}_i^L)$, 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 $q(\mathbf{r}^L, \{\mathbf{G}^l\}_{l=1}^{L-1})$ 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.

Metric	Data	SC	SBM	GP	LAGP	GAE	VGAE	SEAL	G2G	Link-DGP
AUC	**USAir**	74.22 ±3.11	94.85 ±1.14	79.45 ±3.29	82.78 ±2.31	90.26 ±1.41	91.30 ±1.47	96.62 ±0.72	91.26 ±0.56	**97.37** ±0.62
	Router	68.79 ±2.42	85.65 ±1.93	60.84 ±3.43	61.45 ±3.80	74.41 ±1.76	75.84 ±2.69	95.36 ±1.19	73.30 ±2.77	**97.32** ±0.53
	Power	91.78 ±0.61	66.57 ±2.05	83.80 ±1.15	67.17 ±1.85	75.28 ±3.06	79.23 ±1.73	76.88 ±0.75	90.42 ±0.40	**97.72** ±0.70
	Yeast	93.25 ±0.40	91.41 ±0.60	81.30 ±1.14	85.38 ±1.80	93.07 ±0.94	94.15 ±1.05	97.89 ±0.24	95.74 ±0.11	**98.02** ±0.33
	Cora	84.60 ±0.01	63.18 ±2.08	57.33 ±1.78	58.84 ±1.38	91.00 ±0.02	91.40 ±0.01	92.76 ±0.36	90.38 ±0.66	**98.09** ±0.25
	Citeseer	80.50 ±0.01	58.64 ±2.93	61.39 ±2.16	51.28 ±3.43	89.50 ±0.04	90.80 ±0.02	90.59 ±0.21	92.26 ±0.50	**98.72** ±0.34
AP	**USAir**	78.02 ±2.92	95.08 ±1.10	69.82 ±3.15	84.02 ±2.10	91.09 ±1.94	92.43 ±1.75	96.80 ±0.55	91.08 ±0.84	**97.31** ±0.52
	Router	73.53 ±1.47	84.67 ±1.89	58.17 ±3.50	63.51 ±3.98	81.42 ±1.76	82.39 ±2.50	95.32 ±1.47	79.61 ±1.28	**96.85** ±0.51
	Power	91.00 ±0.58	65.48 ±1.85	83.80 ±1.15	67.45 ±1.63	76.17 ±4.15	81.78 ±2.00	80.99 ±0.91	92.42 ±0.40	**97.62** ±0.70
	Yeast	94.63 ±0.56	92.73 ±0.44	74.56 ±2.34	86.12 ±1.76	94.13 ±0.89	95.07 ±0.86	**98.31** ±0.24	96.51 ±0.17	98.28 ±0.26
	Cora	88.50 ±0.00	64.32 ±1.65	58.40 ±1.94	60.86 ±1.52	92.00 ±0.03	92.60 ±0.10	94.04 ±0.29	91.99 ±0.62	**97.73** ±0.48
	Citeseer	80.50 ±0.01	63.29 ±2.74	64.64 ±3.37	52.38 ±4.04	89.90 ±0.05	92.00 ±0.02	92.61 ±0.15	93.01 ±0.62	**98.16** ±0.67

Table 1: Experimental results for the Link-DGP model compared to a number of baselines in terms of area under the ROC curve (**AUC**) and average precision (**AP**). Results are reported with their standard deviation over 10 data splits and the highest mean in each row is highlighted in **bold**.

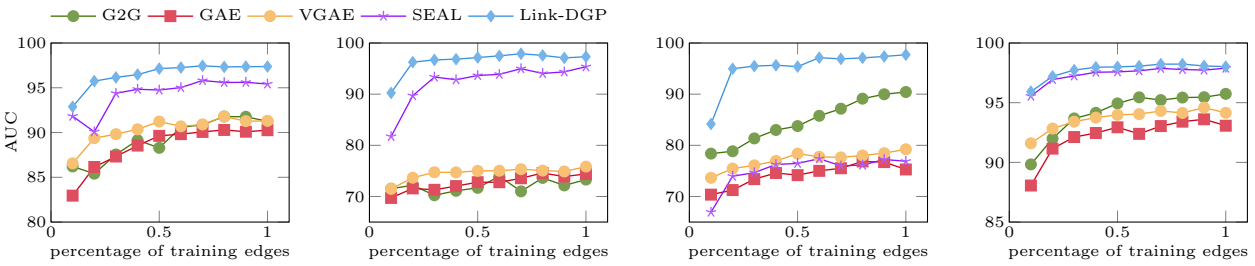
Skip-connections The oversmoothing problem in graph neural networks has demonstrated the importance of decoupling the expressivity of a model, usually specified by its number of layers, and the number of convolutions it applies to the graph signal (Xu et al., 2018; Fey, 2019). This allows increasing the expressiveness of a model by increasing the number of layers without oversmoothing the signal due to an excessive number of convolutions. At the same time, applying stronger smoothing can be achieved without adding additional complexity in the form of another layer. In the deep graph convolutional Gaussian process model, we achieve this by allowing to skip convolutions in each layer. In each Node-GP layer, the Gaussian process units are split into $K_{\max} + 1$ groups with an equal number of units, where $K_{\max}$ is the maximum number of convolutions. A Node-GP in group $k = 0, \dots, K_{\max}$ then performs exactly k convolutions. As a result, units in subsequent layers receive inputs with varying degrees of smoothing. We use automatic relevance determination (ARD) kernels in all layers, such that the model can learn to use outputs of Node-GP units with the most suitable degree of smoothing, thus effectively allowing the model to skip convolutions that restrict the expressiveness of the graph signal too much. As an extreme example and following the leftmost path in the hierarchy in Figure 1 (d), a graph signal that has been subject to zero convolutions can reach the final Link-GP unit, which can then prioritise convolutions using the convolution weights (cf. Equation 5).

Scalability Deep Gaussian process models can be computationally challenging as they require inverting and multiplying large matrices. Using sparse Gaussian processes in each layer and a variational approximation, as described in Sections 2.2 and 4.2 reduces the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(MN^2)$, where N is the number of observations and M is the number of inducing points. Furthermore, it enables optimisation in mini-batches, thus allowing us to operate on multiple smaller matrices rather than one huge matrix. To train with a mini-batch on the graph domain, we need to extract the subgraph that contains all nodes required for performing K graph convolutions, i.e. the K-hop neighbourhood around each edge's incident nodes in the mini-batch. As a consequence, the subgraph size, and thus the size of the matrix that needs to be inverted in one parameter update step, grows exponentially in the exponent K of the convolution matrix: $\mathcal{O}(d_{\max}^K)$. To reduce the computational cost further, we suggest employing neighbourhood sampling, as initially proposed by Hamilton et al. (2017). This gives us full control over the trade-off between computational speed and the amount of neighbourhood information available for each prediction.

5 EXPERIMENTS

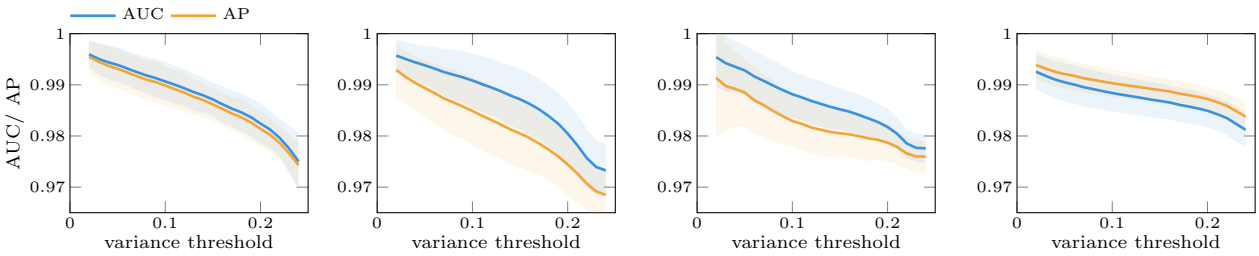
We apply our method on six benchmark data sets with and without node features. USAir (Batagelj and Mrvar, 2006), Router (Spring et al., 2004), Power (Watts and Strogatz, 1998), and Yeast (von Mering et al., 2002) come without node features, whereas Cora and Citeseer (Sen et al., 2008) have node features included. Those data sets were chosen because of their diverse set of properties. They cover both smaller and larger graphs, ones with and without node features, and come from different application domains. A more detailed overview of their properties is given in Appendix B.

Comparison to existing methods We compare the performance of our proposed deep Gaussian process model (Link-DGP) to a number of competitive link prediction methods: spectral clustering (SC), the stochastic block model (SBM) proposed by Airoldi et al. (2008a), a vanilla Gaussian process that operates on the domain of concatenated node features (GP), the link-analysis Gaussian process (LAGP) introduced by Yu and Chu (2008), the graph autoencoder in its non-probabilistic (GAE) and variational (VGAE)



(a) USAir (b) Router (c) Power (d) Yeast

Figure 2: Prediction performance in terms of AUC for different fractions of training edges available.



(a) USAir (b) Router (c) Power (d) Yeast

Figure 3: Prediction performance of the model when rejecting samples with high predictive variance. The prediction of the model is only considered if its predictive variance is below a certain threshold. As the variance threshold increases and fewer low-confidence samples are rejected, we expect the model performance to decrease.

form (Kipf and Welling, 2016), SEAL (Zhang and Chen, 2018), and the graph2gauss (g2g) model (Bojchevski and Günnemann, 2018). Our deep Gaussian process model will use $L = 4$ layers and up to $K = 2$ convolutions. All experiments are performed with the typical setup for link prediction with 10% test edges and under equivalent conditions including for the baselines. The results for SC and the SBM (except for Cora and Citeseer) are as reported by Hasanzadeh et al. (2019). Further details on the experimental setup can be found in Appendix C.

The results of our experiments are shown in Table 1. We find that Link-DGP outperforms all other methods in terms of area under the receiver operating characteristic curve (AUC) on all data sets, often by a large margin. It also performs at least on par with all other methods in terms of average precision (AP) on all data sets. The results demonstrate that Link-DGP is a powerful non-parametric method for link prediction with an inductive bias suitable for a range of domains. The vanilla Gaussian process performs surprisingly well on some of the data sets but there is a clear benefit of added expressivity and inductive bias with Link-DGP. Similarly, Link-DGP performs noticeable better than the shallow link analysis Gaussian process (LAGP), which does not aggregate neighbourhood information. We also perform an ablation study on the number of layers and the effect of skip connections with results reported in Appendix D.

Unlike the baseline models, Link-DGP performs Bayesian inference of the parameters instead of computing point estimates, which comes with the benefit of a partial protection against overfitting and the availability of confidence estimates. On the other hand, this naturally leads to longer training times. However, even for the largest data set, the run time stays below 2 minutes per epoch.

Data efficiency Non-parametric Bayesian methods often tend to perform better compared to parametric models, for example neural networks, under conditions of reduced labelled data. Each parameter in a neural network needs to be learnt using maximum likelihood estimation (MLE), which leads to overfitting, especially when few observations are available. We verify that this assumption holds for Link-DGP, which despite its complexity, only requires relatively few non-variational parameters to be estimated using MLE. For each layer, those are the ARD weights, the convolution weights, and the hyperparameters of the node feature kernel. In Figure 2, we plot the model performance in terms of AUC for different percentages of edges available during training, under otherwise equivalent training conditions.

As expected, model performance tends to drop as less training data is made available. Yet, the Link-DGP model maintains its edge over the baseline models on all data sets for all training set sizes. Moreover, the performance decreases comparatively slowly as the number of training edges decreases, confirming the suitability of Bayesian non-parametric methods for low-data environments.

Quality of confidence estimates A key merit of Bayesian inference using Gaussian process is the availability of predictive variance estimates, i.e. the variance of the distribution $q(y_{ij}) = \int p(y_{ij}|r_{ij}^L)q(r_{ij}^L)\,\mathrm{d}r_{ij}^L$. We examine the suitability of the predictive variance as a measure of certainty in a prediction by allowing the model to reject samples it predicts with high variance (i.e. low certainty). If the uncertainty estimates are well calibrated, performance should increase as we exclude the samples predicted with higher uncertainty. We visualise the results for four data sets in Figure 3 and find, as expected, that as the variance threshold *increases* and therefore *fewer* samples are rejected due to their high predictive uncertainty, the model performance starts to drop to the numbers reported in Table 1. The results for all data sets can be found in Figure 6 of Appendix F.

6 DISCUSSION

We can identify multiple reasons for the performance improvements with Link-DGP. Firstly, it benefits from using both graph structure and node features in a node neighbourhood (compare Link-DGP and LAGP in Table 1). Secondly, we find evidence that computing probabilistic embeddings for nodes greatly improves link prediction performance (compare Link-GP and Link-DGP-2 in Table 3 of Appendix D). Furthermore, the non-parametric nature of the model and the Bayesian inference scheme provide partial protection against overfitting (see Figure 2). Finally, adapting the degree of smoothing is beneficial on certain data sets (compare Link-DGP-4-no-skip and Link-DGP in Table 3 of Appendix D).

In summary, we have proposed a novel Bayesian link prediction model based on deep Gaussian processes that makes use of both the node features and the graph topology. As such, our main contribution is a powerful prior for the topology of attributed graphs, that is suitable for completing a partial graph structure, i.e. predicting missing links. We have demonstrated that it fulfils many requirements we expect from a Bayesian link prediction method: strong predictive performance, high data efficiency, and well calibrated confidence estimates.

Acknowledgements

We would like to thank reviewers for their helpful comments. FLO acknowledges funding from the Huawei Hisilicon Studentship at the Department of Computer Science and Technology of the University of Cambridge.

References

C. Aicher, A. Z. Jacobs, and A. Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 2014.

E. M. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, 2008a.

E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, pages 1981–2014, 2008b.

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

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

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

W. Chu, V. Sindhwani, Z. Ghahramani, and S. S. Keerthi. Relational learning with gaussian processes. In *Advances in Neural Information Processing Systems 19*, pages 289–296, 2007.

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

T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852, 2016.

P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, page 290, 1959.

M. Fey. Just jump: Dynamic neighborhood aggregation in graph neural networks, 2019. arXiv preprint arXiv: 1904.04849.

A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 855–864, 2016.

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

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

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

J. Hensman, A. G. de G. Matthews, and Z. Ghahramani. Scalable variational gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Y. Kashinskaya, E. Samosvat, and A. Artikov. Spring-electrical models for link prediction. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 708–716, 2019.

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

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

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

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

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

J. Klicpera, A. Bojchevski, and S. Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.

M. Lázaro-Gredilla and A. Figueiras-Vidal. Inter-domain gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems 22*, pages 1087–1095, 2009.

Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, page 319, 1999.

C. Lei and J. Ruan. A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics*, pages 355–364, 2012.

N. Li, W. Li, J. Sun, Y. Gao, Y. Jiang, and S.-T. Xia. Stochastic deep gaussian processes over graphs. In *Advances in Neural Information Processing Systems*, 2020.

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

J. Lloyd, P. Orbanz, Z. Ghahramani, and D. M. Roy. Random function priors for exchangeable arrays with applications to graphs and relational data. In *Advances in Neural Information Processing Systems*, 2012.

Y. C. Ng, N. Colombo, and R. Silva. Bayesian semi-supervised learning with graph gaussian processes. In *Advances in Neural Information Processing Systems 31*, pages 1683–1694, 2018.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

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

Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

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

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

R. Silva, W. Chu, and Z. Ghahramani. Hidden common cause relations in relational learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1345–1352, 2008.

N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, pages 2–16, 2004.

J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015.

M. van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems 30*, pages 2849–2858, 2017.

C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, pages 399–403, 2002.

I. Walker and B. Glocker. Graph convolutional Gaussian processes. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6495–6504, 2019.

P. Wang, B. Xu, Y. Wu, and X. Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, pages 1–38, 2015.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, pages 440–442, 1998.

F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6861–6871, 2019a.

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

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

K. Yu and W. Chu. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems 20*, pages 1657–1664, 2008.

X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S. M. Lin, W. Zhang, P. Zhang, and H. Sun. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, pages 1241–1251, 2019.

M. Zhang and Y. Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–583, 2017.

M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems 31*, pages 5165–5175, 2018.

Y.-C. Zhi, Y. C. Ng, and X. Dong. Gaussian processes on graphs via spectral kernel learning, 2020. arXiv preprint arXiv: 2006.07361.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, pages 912–919, 2003a.

X. Zhu, J. D. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical report, Carnegie Mellon University, Computer Science Department, 2003b.

Supplementary Material: Bayesian Link Prediction with Deep Graph Convolutional Gaussian Processes

A KERNEL ANALYSIS

To obtain a better understanding of the effect of the simplified graph convolutions applied to the Gaussian process, we examine the prior covariance of $\mathbf{g}$ between two nodes for the case that all convolution weights have been set to 1:

$$[\tilde{\mathbf{S}}^K \mathbf{K} \tilde{\mathbf{S}}^K]_{ij} = \sum_{\substack{k \in \mathcal{N}^K(i) \\ \cup\{i\}}} \sum_{\substack{l \in \mathcal{N}^K(j) \\ \cup\{j\}}} [\tilde{\mathbf{S}}^K]_{ik} [\tilde{\mathbf{S}}^K]_{lj} K_{kl}. \tag{14}$$

Here, $\mathcal{N}^K(i)$ refers to the K-hop neighbourhood of node i. We note that given the definition of the convolution matrix (cf. Equation 5), the coefficients $[\tilde{\mathbf{S}}^K]_{ik}$ and $[\tilde{\mathbf{S}}^K]_{lj}$ lie in the interval $[0, 1]$. Furthermore, for a fixed K, the j^{th} entry of the i^{th} row of $\tilde{\mathbf{S}}^K$ is non-zero if and only if j is in the K-hop neighbourhood of i. Therefore, for larger K, more entries of $\tilde{\mathbf{S}}^K$ will be non-zero, as the size of the neighbourhood increases, yet every individual entry will be smaller because elements in $[0, 1]$ are being multiplied. As a result, as we increase K, more but smaller terms are summed in Equation 14, leading to the covariance to be spread across neighbourhoods of different sizes more equally. We confirm this empirically by choosing a random node in the input graph as the central node and plotting the average covariance of nodes at different geodesic distances. We observe that as K is increased, the differences between the average covariance values start to shrink. The result is visualised in Figure 4.

We expect a graph function for which the covariance between two distant nodes is higher to vary less from node to node compared to a function that has low covariance for distant nodes. Hence, we expect the function to be smoother as measured by the Dirichlet norm

$$\|\mathbf{g}\|_{\mathcal{G}}^2 = \frac{1}{2} \sum_{i,j=1}^{N} a_{ij} (g_i - g_j)^2 = \mathbf{g}^\top \mathbf{L} \mathbf{g}, \tag{15}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian. This agrees with the result of Wu et al. (2019a), who have shown that simplified graph convolutions act as a low pass filter, thus smoothing the graph signal. In Figure 5, we plot the average Dirichlet norm for functions sampled from the described Gaussian process prior for varying K. As expected, the smoothness of the sampled functions increases for larger K. We have found these observations to generalise well across data sets.

B DETAILED DATA SET INFORMATION

Our method is evaluated on six data sets: the USAir (Batagelj and Mrvar, 2006) data set of US airline connections, the Router (Spring et al., 2004) data set describing Internet Service Provider topology, the Power (Watts and Strogatz, 1998) data set describing the western US grid network, the Yeast (von Mering et al., 2002) data set containing the protein-protein interaction network for yeast, and the Cora and Citeseer data sets (Sen et al., 2008) containing citation networks of scientific publications. More detailed statistics about the data sets can be found in Table 2.

C DETAILED EXPERIMENTAL SETUP

We use the typical technique for computing the data set split into training and test set by randomly selecting 10% of edges as test edges and removing them from the graph. We then randomly select an equal number of node pairs that are not connected by an edge as negative test samples. The remaining edges are used for the

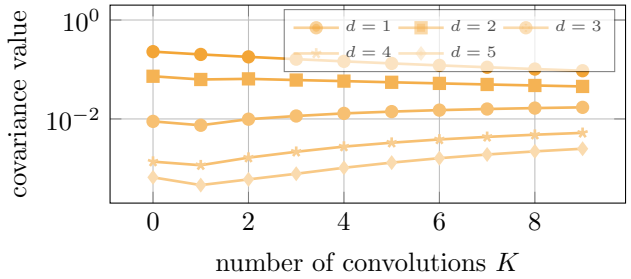


Figure 4: Average covariances between nodes of varying geodesic distance. We randomly pick a node i in the graph of the Yeast data set (for details, see Section 5.1). We then construct 5 disjoint sets of nodes that have geodesic distance of exactly $d = 1, \ldots, 5$ from i and compute the covariance between node i and the nodes in each set, averaged over the nodes within the same set. We plot this mean covariance value for different numbers of convolutions K. We use an RBF-kernel as the node feature kernel with lengthscale and variance set to 1.0. We find that as K increases, the mean covariance values grow closer together, indicating that the covariance spreads more equally over the graph.

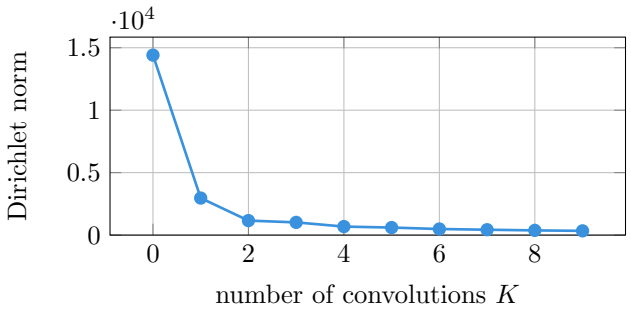


Figure 5: Average Dirichlet norm of 5,000 functions sampled from the graph convolutional Gaussian process prior for varying number of convolutions K. We use the graph and `node2vec` features of the Yeast data set (for details, see Section 5.1) and an RBF-kernel as the node feature kernel. Its lengthscale and variance are set to 1.0. For larger K, the average Dirichlet norm decreases, indicating that the sampled functions are smoother.

Data	# nodes	# edges	avg. deg.	node feat.
USAir	332	2,126	12.81	no
Router	5,022	6,258	2.49	no
Power	4,941	6,594	2.67	no
Yeast	2,375	11,693	9.85	no
Cora	2,708	5,429	4.01	yes
Citeseer	3,327	4,732	2.84	yes

Table 2: Description of the data sets used in our experiments including the number of nodes, number of edges, average node degree, and availability of node features.

training set and we again select and equal number of pairs of unconnected nodes as negative training samples. Note that we do not require a validation set as model selection will be performed based on the maximum ELBO (cf. Equation 14) achieved on the training set. For the data sets that come without node features (cf. Table 2), we generate them using the `node2vec` embedding generation algorithm introduced by Grover and Leskovec (2016) using the same setup as in Zhang and Chen (2018).

For all models and data sets we report the mean and standard deviation of the model performance for 10 different random seeds. Seeds influence the data set split and the parameter initialisation in case of the neural network baselines.

All experiments were run on a shared cluster of Nvidia P100 GPUs with 16GB of GPU memory each.

SBM For the stochastic block model (SBM), the results for all data sets except Cora and Citeseer were reported by Hasanzadeh et al. (2019) who use the code provided by Aicher et al. (2014) with 12 latent groups. We follow the same setup to obtain results for Cora and Citeseer.

Neural network baselines For the variational graph auto-encoder and the graph2gauss model, we use the implementation provided by the authors with $K = 2$ convolutions and otherwise 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 Link-GP The following training settings apply to the single-layer Link-GP as evaluated in the ablation study in Appendix D. In all, experiments, we set the maximum number of convolutions to $K = 2$. A random inducing graph with $|\bar{V}| = \frac{|V|}{8}$ nodes and $|\bar{E}| = 2|\bar{V}|$ edges is drawn from an Erdős-Rényi model (Erdös and Rényi, 1959). The pseudo-inputs are initialised with K-means on the node features of the training data set. For the node feature kernel, we use a constant mean function and an radial basis function (RBF) kernel with automatic relevance determination (ARD), defined as

$$k(\mathbf{x}, \mathbf{x}') = \nu \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{l_d^2}\right). \tag{16}$$

Its variance ν is initialised to 1.0 and its lengthscales l_d are initialised to either 1.0 or 2.0 and the results of the model achieving a higher ELBO on the training set are reported. The convolution weights λ_1 and λ_2 are initialised to 0.5 and 0.3 respectively. For the data sets that come without node features, we use `node2vec` embeddings of size 128, following Zhang and Chen (2018). For parameter optimisation, we use the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.001. We train our models for up to 1000 epochs with a batch size of 128 edges and stop training early if there is no improvement in ELBO over 100 epochs.

Link-DGP For hyperparameters and training settings that the single-layer Link-GP and the deep Link-DGP have in common, we use the same values, except where noted otherwise. In particular, the variance ν and lengthscales l_d of all kernels are always initialised to 1.0. Each hidden layer of the Link-DGP consists of 33 Gaussian process units. A single sample from $q(\mathbf{f}_i^L)$ is used to approximate the likelihood term of the ELBO (cf. Equation 14) during training and 50 during testing. Further model details follow the approach by Salimbeni and Deisenroth (2017). In particular, the mean functions of hidden layers are initialised to linear maps $m(\mathbf{X}) = \mathbf{X}\mathbf{W}$. If the number of input dimensions of the layer is less or equal to the number of output dimensions, the (potentially zero-padded) identity matrix is used for $\mathbf{W}$. If the number of input dimensions is greater than the number of 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.

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.

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 14. 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. We use a learning rate of 0.01 during all of training. For improved training stability we clip the gradients during the update step to the average gradient magnitude during the first 20 epochs.

Vanilla Gaussian process For the Gaussian process operating on the concatenated node features we use the same training setting as for the single-layer Link-GP. The Gaussian process prior uses an RBF kernel with automatic relevance determination and with lengthscales initialised to 1.0. The number of inducing points matches the number of inducing nodes used for the Link-GP. The pseudo-inputs are initialised using K-means on the vectors obtained when concatenating each pair of features of the nodes incident to each training edge.

D ABLATION STUDY

Number of Gaussian process layers We compare the performance of a single-layer Link-GP model defined in Equation 12 with the deep Gaussian process Link-DGP model defined in Section 4.3 with a varying number of Node-GP layers. The Link-DGP-3 model has 2 Node-GP layers followed by a final Link-GP layer. The training settings for the single-layer Link-GP model closely follow those for its deep Gaussian process equivalent and are described in detail in Section C. All deep Gaussian process models use the same training settings, naturally with the exception of the number of layers.

The results of the comparison can be found in Table 3. We find that both in terms of AUC and AP, deeper models tend to perform better. In particular, the biggest improvement is usually observed when moving from the

Metric	Data	Link-GP	Link-DGP-2	Link-DGP-3	Link-DGP-4 no-skip	Link-DGP-4
AUC	**USAir**	96.58 ±0.78	97.50 ±0.64	97.53 ±0.67	**96.76** ±0.83	97.37 ±0.62
	Router	81.79 ±1.69	96.02 ±0.73	96.92 ±1.05	96.15 ±2.01	**97.32** ±0.53
	Power	83.31 ±1.07	96.00 ±1.14	97.01 ±1.26	97.70 ±0.78	**97.72** ±0.70
	Yeast	97.09 ±0.32	97.56 ±0.27	97.77 ±0.16	97.74 ±0.28	**98.02** ±0.33
	Cora	89.37 ±0.94	97.25 ±0.85	97.98 ±0.16	98.06 ±0.09	**98.09** ±0.25
	Citeseer	67.54 ±2.10	97.73 ±0.31	99.07 ±0.28	**99.21** ±0.31	98.72 ±0.34
AP	**USAir**	97.28 ±0.57	97.28 ±0.68	97.44 ±0.59	**97.73** ±0.91	97.31 ±0.52
	Router	86.34 ±1.22	95.46 ±0.64	96.34 ±1.18	95.59 ±1.66	**96.85** ±0.51
	Power	86.59 ±0.74	96.02 ±1.00	96.86 ±1.19	97.57 ±0.73	**97.62** ±0.70
	Yeast	97.82 ±0.22	97.97 ±0.24	98.14 ±0.18	98.10 ±0.21	**98.28** ±0.26
	Cora	88.77 ±1.61	96.38 ±1.29	97.80 ±0.17	97.47 ±0.55	**97.73** ±0.48
	Citeseer	70.18 ±1.53	97.42 ±0.57	98.78 ±0.63	**98.98** ±0.56	98.16 ±0.67

Table 3: Performance of the single-layer Link-GP and the multi-layer Link-DGP with varying numbers of layers and with and without skip-connections in terms of area under the ROC curve (**AUC**) and average precision (**AP**). All results are reported with their standard deviation over 10 runs and the results with the highest mean in each column are highlighted in **bold**.

shallow Gaussian process model to the 2-layer deep Gaussian process model. For example, for the Power data set, the Link-GP model achieves an AUC of 83.31, whereas the Link-DGP-2 model achieves 96.00, which increases to 97.72 as more layers are added. The results indicate that using a hierarchy of Gaussian processes drastically improves the inductive bias of the model and additional layers tend to improve the performance further.

Non-convolutional Deep Gaussian process We also evaluated a deep Gaussian process model operating on concatenated node features, but found that it fails to converge, which we conjecture is due to its weak inductive bias and lack of suitable constraints on the kernel.

Removing skip-connections We also evaluate the Link-DGP-4 model without skip connections between layers. We still use the same number of Gaussian process units in each layer however all units perform the full number of K convolutions. The results in Table 3 indicate that Link-DGP with skip-connections always performs as least as good as its variant without skip-connections except for the Citeseer data set. On the Router data set the drop in AUC from removing skip-connections is particularly pronounced. The ablation study suggests that the full Link-DGP model achieves overall the most consistent strong performance across data sets.

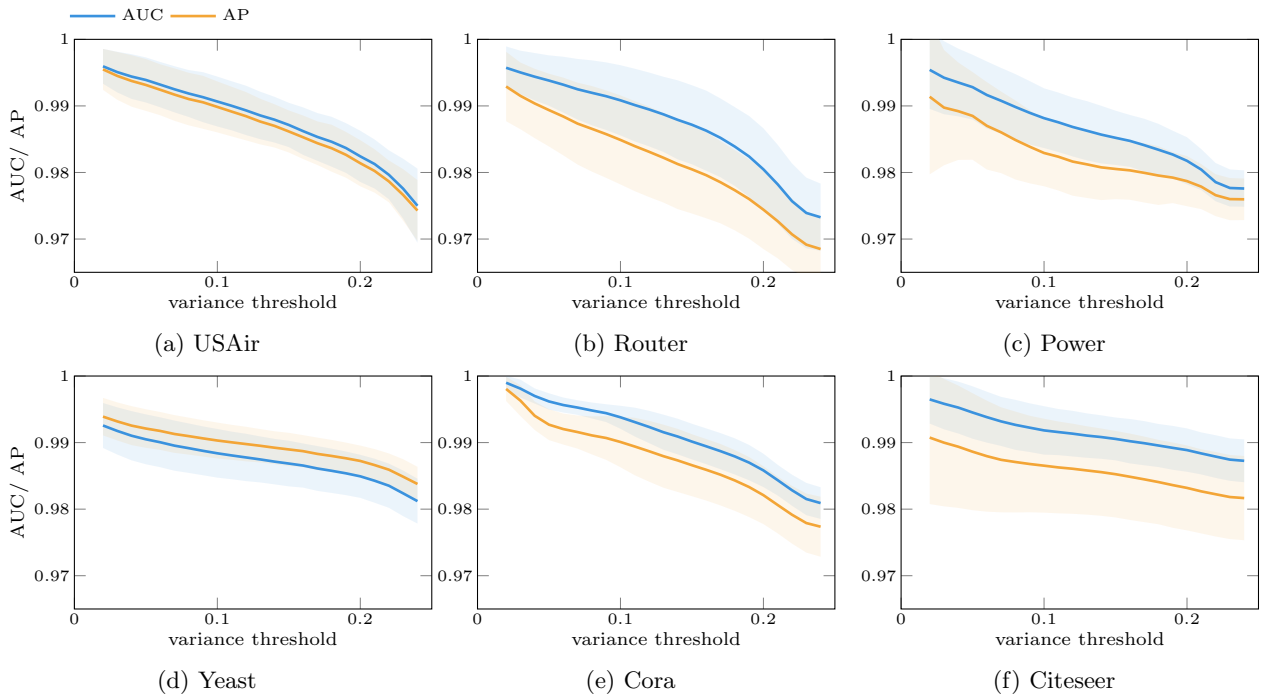
E BROADER IMPACT

Societal impact We present a methodological study that is not directly linked to a specific application. This means its results can also be applied in a range of real-world scenarios with dangerous negligence or even malicious intent. This is particularly precarious when making predictions on a network of people for example to infer connections between people, which could have negative consequences for them.

Environmental impact We also note that Gaussian processes perform Bayesian inference to obtain well calibrated confidence estimates and partial protection against overfitting rather than compute point estimates of parameters. This naturally has a higher computational demand, which could result in higher energy usage and potentially higher CO2 emissions.

F ADDITIONAL EXPERIMENTAL RESULTS

The results of the experiments studying the quality of the confidence estimates of the Link-DGP model as described in Section 5 are shown in Figure 6.



(a) USAir (b) Router (c) Power

(d) Yeast (e) Cora (f) Citeseer

Figure 6: Prediction performance of the model when rejecting samples with high predictive variance. The prediction of the model is only considered if its predictive variance is below a certain threshold. As the variance threshold increases and fewer low-confidence samples are rejected, we expect the model performance to decrease.

G VARIATIONAL APPROXIMATION FOR LINK-DGP

In this Section, we provide more details on the variational approximation of the posterior briefly described in Section 4.3. We assume the joint distribution of our models factorises as

$$p\left(\mathbf{y}, \mathbf{r}, \{\mathbf{G}^l\}_{l=1}^{L-1}\right) = \underbrace{\left[\prod_{e_{ij}\in\mathcal{E}} p(y_{ij} \,|\, r_{ij})\right]}_{\text{likelihood } p(\mathbf{y}\,|\,\mathbf{r})} \underbrace{p(\mathbf{r} \,|\, \mathbf{G}^{L-1}) \prod_{l=1}^{L-1} p(\mathbf{G}^l \,|\, \mathbf{G}^{l-1})}_{\text{Deep GP prior}}, \tag{17}$$

where $\mathbf{G}^0 = \mathbf{X} \in \mathbb{R}^{N\times D}$ and $\mathbf{y} = [y_{ij}, \ldots]^\top \in \mathbb{R}^E$, $\mathbf{r} = [r_{ij}, \ldots]^\top \in \mathbb{R}^E$, $\mathbf{G}^l \in \mathbb{R}^{N\times K^l}$ and $K^0 = D$. Here, $p(y_{ij} \,|\, r_{ij}) = \text{Ber}(y_{ij} \,|\, r_{ij})$ is a Bernoulli likelihood, $\mathbf{r}$ is assigned a Link-GP prior as defined in Equation 11, and $\mathbf{G}^l_k$ is assigned a Node-GP prior as defined in Equation 10. For scalability, we introduce inducing points and add them to the joint, resulting in the extended joint

$$p\left(\mathbf{y}, \mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right) = p(\mathbf{y} \,|\, \mathbf{r}) p(\mathbf{r} \,|\, \mathbf{u}^L, \mathbf{G}^{L-1}) p(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{G}^l \,|\, \mathbf{U}^l, \mathbf{G}^{l-1}) p(\mathbf{U}^l), \tag{18}$$

where $\mathbf{u}^L$ follows the same Link-GP prior as $\mathbf{r}$ and $\mathbf{U}^l$ follows the same Node-GP prior as $\mathbf{G}^l$. The posterior distribution over the latent variables has no analytical solution and we therefore approximate it with a variational posterior, which we choose as

$$q\left(\mathbf{r}, \mathbf{u}^L \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right) = p(\mathbf{r} \,|\, \mathbf{u}^L, \mathbf{G}^{L-1}) q(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{G}^l \,|\, \mathbf{U}^l, \mathbf{G}^{l-1}) q(\mathbf{U}^l) \tag{19}$$

with $q(\mathbf{u}^L) = \mathcal{N}(\mathbf{m}^L, \mathbf{S}^L)$ and $q(\mathbf{U}) = \prod_{k=1}^K \mathcal{N}(\mathbf{m}^{l,k}, \mathbf{S}^{l,k})$ with variational means $\mathbf{m}^l \in \mathbb{R}^{M^l}$ and variational covariances $\mathbf{S}^l \in \mathbb{R}^{M^l \times M^l}$, where M^l is the number of inducing points in layer l. This setup allows us to find the

evidence lower bound (ELBO) to the intractable marginal likelihood

$$
\begin{aligned}
\log p(\mathbf{y}) &= \log \iiiint p\left(\mathbf{y}, \mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l, \}_{l=1}^{L-1}\right) \mathrm{d}\mathbf{r}\, \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{G}^l, \mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&\geq \iiiint q\left(\mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right) \log \frac{p\left(\mathbf{y}, \mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right)}{q\left(\mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right)} \mathrm{d}\mathbf{r}\, \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{G}^l, \mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&= \iiiint q\left(\mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right) \\
&\qquad \log \frac{p(\mathbf{y} \mid \mathbf{r}) \cancel{p(\mathbf{r} \mid \mathbf{u}^L, \mathbf{G}^{L-1})} p(\mathbf{u}^L) \prod_{l=1}^{L-1} \cancel{p(\mathbf{G}^l \mid \mathbf{U}^l, \mathbf{G}^{l-1})} p(\mathbf{U}^l)}{\cancel{p(\mathbf{r} \mid \mathbf{u}^L, \mathbf{G}^{L-1})} q(\mathbf{u}^L) \prod_{l=1}^{L-1} \cancel{p(\mathbf{G}^l \mid \mathbf{U}^l, \mathbf{G}^{l-1})} q(\mathbf{U}^l)} \mathrm{d}\mathbf{r}\, \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{G}^l, \mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&= \iint q\left(\mathbf{r}, \{\mathbf{G}^l\}_{l=1}^{L-1}\right) \log p(\mathbf{y} \mid \mathbf{r})\, \mathrm{d}\mathbf{r} \left\{\mathrm{d}\mathbf{G}^l\right\}_{l=1}^{L-1} \\
&\qquad + \iint q(\mathbf{u}^L) \prod_{l=1}^{L-1} q(\mathbf{U}^l) \log \frac{p(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{U}^l)}{q(\mathbf{u}^L) \prod_{l=1}^{L-1} q(\mathbf{U}^l)} \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&= \mathbb{E}_{q(\mathbf{r})}\left[\log p(\mathbf{y} \mid \mathbf{r})\right] - \mathrm{KL}[q(\mathbf{u}^L) \parallel p(\mathbf{u}^L)] - \sum_{l=1}^{L-1} \mathrm{KL}[q(\mathbf{U}^L) \parallel p(\mathbf{U}^L)].
\end{aligned} \tag{20}
$$

The marginalised variational distribution, which is required to compute the likelihood term of the ELBO, is given by

$$
\begin{aligned}
q\left(\mathbf{r}, \{\mathbf{G}^l\}_{l=1}^{L-1}\right) &= \iint q\left(\mathbf{r}, \mathbf{u}^L, \{\mathbf{G}^l, \mathbf{U}^l\}_{l=1}^{L-1}\right) \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&= \iint p(\mathbf{r} \mid \mathbf{u}^L, \mathbf{G}^{L-1}) q(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{G}^l \mid \mathbf{U}^l, \mathbf{G}^{l-1}) q(\mathbf{U}^l)\, \mathrm{d}\mathbf{u}^L \left\{\mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1} \\
&= \int p(\mathbf{r} \mid \mathbf{u}^L, \mathbf{G}^{L-1}) q(\mathbf{u}^L)\, \mathrm{d}\mathbf{u}^L \cdot \prod_{l=1}^{L-1} \int p(\mathbf{G}^l \mid \mathbf{U}^l, \mathbf{G}^{l-1}) q(\mathbf{U}^l) \left\{\mathrm{d}\mathbf{U}^l\right\}_{l=1}^{L-1}.
\end{aligned}
$$

We continue deriving expressions for the two remaining factors separately. For the first one, we obtain

$$
\begin{aligned}
&\int p(\mathbf{r} | \mathbf{u}^L, \mathbf{G}^{L-1}) q(\mathbf{u}^L)\, \mathrm{d}\mathbf{u}^L \\
&= \int \mathcal{N}\big(\mathbf{r} | m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{u}^L - m(\mathbf{Z}^{L-1})), \\
&\qquad\qquad k(\mathbf{G}^{L-1}, \mathbf{G}^{L-1}) - \alpha(\mathbf{G}^{L-1})^\top k(\mathbf{Z}^{L-1}, \mathbf{Z}^{L-1}) \alpha(\mathbf{G}^{L-1})\big) \\
&\qquad \cdot \mathcal{N}\left(\mathbf{u}^L | \mathbf{m}^L, \mathbf{S}^L\right) \mathrm{d}\mathbf{u}^L \\
&= \int \mathcal{N}\big(\mathbf{r} | m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{u}^\mathbf{r} - m(\mathbf{Z}^{L-1})), \\
&\qquad\qquad k(\mathbf{G}^{L-1}, \mathbf{G}^{L-1}) - \alpha(\mathbf{G}^{L-1})^\top k(\mathbf{Z}^{L-1}, \mathbf{Z}^{L-1}) \alpha(\mathbf{G}^{L-1})\big) \\
&\qquad \cdot \mathcal{N}\big(m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{u}^\mathbf{r} - m(\mathbf{Z}^{L-1})) | \\
&\qquad\qquad m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{m} - m(\mathbf{Z}^{L-1})), \\
&\qquad\qquad \alpha(\mathbf{G}^{L-1})^\top \mathbf{S}^L \alpha(\mathbf{G}^{L-1})\big)\, \mathrm{d}\mathbf{u}^\mathbf{r} \\
&= \mathcal{N}\big(\mathbf{r} | m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{m} - m(\mathbf{Z}^{L-1})), \\
&\qquad\qquad k(\mathbf{G}^{L-1}, \mathbf{G}^{L-1}) - \alpha(\mathbf{G}^{L-1})^\top k(\mathbf{Z}^{L-1}, \mathbf{Z}^{L-1}) \alpha(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top \mathbf{S}^L \alpha(\mathbf{G}^{L-1})\big) \\
&= \mathcal{N}\big(\mathbf{r} | \underbrace{m(\mathbf{G}^{L-1}) + \alpha(\mathbf{G}^{L-1})^\top (\mathbf{m} - m(\mathbf{Z}^{L-1}))}_{:= \tilde{\mu}^L(\mathbf{G}^{L-1})}, \\
&\qquad\qquad \underbrace{k(\mathbf{G}^{L-1}, \mathbf{G}^{L-1}) - \alpha(\mathbf{G}^{L-1})^\top (k(\mathbf{Z}^{L-1}, \mathbf{Z}^{L-1}) - \mathbf{S}^L) \alpha(\mathbf{G}^{L-1})}_{:= \tilde{\mathbf{\Sigma}}^L(\mathbf{G}^{L-1})}\big),
\end{aligned} \tag{21}
$$

where $\alpha(\mathbf{G}_i^{L-1}) = k(\mathbf{Z}^{L-1}, \mathbf{Z}^{L-1})^{-1} k(\mathbf{Z}^{L-1}, \mathbf{G}_i^{L-1})$ and we colour-coded the mean and covariance of each normal distribution for visual clarity. To get from line 3 to line 4 we have used the fact that

$$\int \mathcal{N}(x|\mu_1, \Sigma_1)\mathcal{N}(\mu_1|\mu_2, \Sigma_2)\, \mathrm{d}\mu_1 = \mathcal{N}(x|\mu_2, \Sigma_1 + \Sigma_2). \tag{22}$$

Analogously, a similar expression can be found for the second factor

$$\begin{aligned}
&\prod_{l=1}^{L-1} \int p(\mathbf{G}^l \,|\, \mathbf{U}^l, \mathbf{G}^{l-1}) q(\mathbf{U}^l) \left\{ \mathrm{d}\mathbf{U}^l \right\}_{l=1}^{L-1} \\
&\quad = \prod_{l=1}^{L-1} \mathcal{N}\big(\mathbf{G}^l | \underbrace{m(\mathbf{G}^{l-1}) + \alpha(\mathbf{G}^{l-1})^\top (\mathbf{m} - m(\mathbf{Z}^{l-1}))}_{:= \tilde{\mu}^l(\mathbf{G}^{l-1})}, \\
&\qquad\qquad \underbrace{k(\mathbf{G}^{l-1}, \mathbf{G}^{l-1}) - \alpha(\mathbf{G}^{l-1})^\top (k(\mathbf{Z}^{l-1}, \mathbf{Z}^{l-1}) - \mathbf{S}^l)\alpha(\mathbf{G}^{l-1})}_{:= \tilde{\mathbf{\Sigma}}^l(\mathbf{G}^{l-1})} \big)
\end{aligned} \tag{23}$$

and we therefore obtain the marginalised variational distribution

$$q\left(\mathbf{r}, \left\{\mathbf{G}^l\right\}_{l=1}^{L-1}\right) = \mathcal{N}\left(\mathbf{r} \,|\, \tilde{\mu}^L(\mathbf{G}^{L-1}), \tilde{\mathbf{\Sigma}}^L(\mathbf{G}^{L-1})\right) \prod_{l=1}^{L-1} \mathcal{N}\left(\mathbf{G}^l \,|\, \tilde{\mu}^l(\mathbf{G}^{l-1}), \tilde{\mathbf{\Sigma}}^l(\mathbf{G}^{l-1})\right). \tag{24}$$

The first term in the ELBO in Equation 20 is intractable and, as proposed by Salimbeni and Deisenroth (2017), we therefore opt to approximate it by sampling from the further marginalised variational distribution $q(\mathbf{r})$. This can be achieved via the reparameterisation trick: For a sample from $q(\mathbf{G}_i^l)$, we draw a sample $\epsilon_i^l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and transform it to a sample $\hat{\mathbf{G}}_i^l \sim q(\mathbf{G}_i^l)$ via

$$\hat{\mathbf{G}}_i^l = \tilde{\mu}^{\mathbf{l}}(\mathbf{G}_i^{l-1}) + \epsilon_i^l \odot \tilde{\mathbf{\Sigma}}^{\mathbf{l}}(\mathbf{G}_i^{l-1}). \tag{25}$$

To obtain samples $\tilde{\mathbf{r}} \sim q(\mathbf{r})$, this step is repeated for each layer in the deep Gaussian process, propagating samples through the hierarchy.