RESEARCH   **Open Access**

Joint embedding of structure and features via graph convolutional networks

Sébastien Lerique[1]* , Jacob Levy Abitbol[1] and Márton Karsai[1,2]

*Correspondence:
sebastien.lerique@normalesup.org
[1]Univ Lyon, ENS de Lyon, Inria,
CNRS, UCB Lyon 1, LIP UMR 5668,
IXXI, 69007 Lyon, France
Full list of author information is
available at the end of the article

Abstract

The creation of social ties is largely determined by the entangled effects of people's similarities in terms of individual characters and friends. However, feature and structural characters of people usually appear to be correlated, making it difficult to determine which has greater responsibility in the formation of the emergent network structure. We propose *AN2VEC*, a node embedding method which ultimately aims at disentangling the information shared by the structure of a network and the features of its nodes. Building on the recent developments of Graph Convolutional Networks (GCN), we develop a multitask GCN Variational Autoencoder where different dimensions of the generated embeddings can be dedicated to encoding feature information, network structure, and shared feature-network information. We explore the interaction between these disentangled characters by comparing the embedding reconstruction performance to a baseline case where no shared information is extracted. We use synthetic datasets with different levels of interdependency between feature and network characters and show (i) that shallow embeddings relying on shared information perform better than the corresponding reference with unshared information, (ii) that this performance gap increases with the correlation between network and feature structure, and (iii) that our embedding is able to capture joint information of structure and features. Our method can be relevant for the analysis and prediction of any featured network structure ranging from online social systems to network medicine.

Keywords: Network embedding, Attributed networks, Graph, Convolutional networks, Autoencoders, Feature-network dependency

Although it is relatively easy to obtain the proxy social network and various individual features for users of online social platforms, the combined characterisation of these types of information is still challenging our methodology. While current approaches have been able to approximate the observed marginal distributions of node and network features separately, their combined consideration was usually done via summary network statistics merged with otherwise independently built feature sets of nodes. However, the entanglement between structural patterns and feature similarities appears to be fundamental to a deeper understanding of network formation and dynamics. The value of this joint information then calls for the development of statistical tools for the learning of combined representation of network and feature information and their dependencies.

The formation of ties and mesoscopic structures in online social networks is arguably determined by several competing factors. Considering only network information, neighbour similarities between people are thought to explain network communities, where triadic closure mechanisms (Kumpula et al. 2007; Kossinets and Watts 2006) induce ties between peers with larger fractions of common friends (Granovetter 1977). Meanwhile, random bridges (Kossinets and Watts 2006) are built via focal closure mechanisms, optimising the structure for global connectedness and information dissemination. At the same time, people in the network can be characterised by various individual features such as their socio-demographic background (Leo et al. 2016; Levy Abitbol et al. 2018b), linguistic characters (Gumperz 2009; Levy Abitbol et al. 2018a; Levy Abitbol et al. 2018b), or the distributions of their topics of interests (Cataldi et al. 2010; Hours et al. 2016), to mention just a few. Such features generate homophilic tie creation preferences (McPherson et al. 2001; Kossinets and Watts 2009), which induce links with higher probability between similar individuals, whom in turn form feature communities of shared interest, age, gender, or socio-economic status, and so on (Leo et al. 2016; Shrum et al. 1988). Though these mechanisms are not independent and lead to correlations between feature and network communities, it is difficult to define the causal relationship between the two: first, because simultaneously characterising similarities between multiple features and a complex network structure is not an easy task; second, because it is difficult to determine, which of the two types of information, features or structure, is driving network formation to a greater extent. Indeed, we do not know what fraction of similar people initially get connected through homophilic tie creation, versus the fraction that first get connected due to structural similarities before influencing each other to become more similar (La Fond and Neville 2010; Aral et al. 2009).

Over the last decade popular methods have been developed to characterise structural and feature similarities and to identify these two notions of communities. The detection of network communities has been a major challenge in network science with various concepts proposed (Blondel et al. 2008; Rosvall et al. 2009; Peixoto 2014) to solve it as an unsupervised learning task (Fortunato 2010; Fortunato and Hric 2016). Commonly, these algorithms rely solely on network information, and their output is difficult to cross-examine without additional meta-data, which is usually disregarded in their description. On the other hand, methods grouping similar people into feature communities typically ignore network information, and exclusively rely on individual features to solve the problem as a data clustering challenge (Jain 2010; Gan et al. 2007). Some semi-supervised learning tasks, such as link prediction, may take feature and structural information simultaneously into account, but only by enriching individual feature vectors with node-level network characteristics such as degree or local clustering (Liben-Nowell and Kleinberg 2007; Lü and Zhou 2011; Hours et al. 2016). Methods that would take higher order network correlations and multivariate feature information into account at the same time are still to be defined. Their development would offer huge potential in understanding the relation between individuals' characters, their social relationships, the content they are engaged with, and the larger communities they belong to. This would not only provide us with deeper insight about social behaviour, it would give us predictive tools for the emergence of network structure, individual interests and behavioural patterns.

In this paper we propose a contribution to solve this problem by developing a joint feature-network embedding built on multitask Graph Convolutional Networks (Kipf

and Welling 2016a; Bruna et al. 2013; Hamilton et al. 2017; Ying et al. 2018) and Variational Autoencoders (GCN-VAE) (Kingma and Welling 2013; Rezende et al. 2014; Kipf and Welling 2016b; Wang et al. 2016; Zhu et al. 2018), which we call the Attributed Node to Vector method (AN2VEC). In our model, different dimensions of the generated embeddings can be dedicated to encode feature information, network structure, or shared feature-network information separately. Unlike previous embedding methods dealing with features (Gao and Huang 2018; Tran 2018; Shen et al. 2018; Bojchevski and Günnemann 2017), this interaction model (Aiken et al. 1994) allows us to explore the dependencies between the disentangled network and feature information by comparing the embedding reconstruction performance to a baseline case where no shared information is extracted. Using this method, we can identify an optimal reduced embedding, which indicates whether combined information coming from the structure and features is important, or whether their non-interacting combination is sufficient for reconstructing the featured network.

In practice, as this method solves a reconstruction problem, it may give important insights about the combination of feature- and structure-driven mechanisms which determine the formation of a given network. As an embedding, it is useful to identify people sharing similar individual and structural characters. And finally, by measuring the optimal overlap between feature– and network-associated dimensions, it can be used to verify network community detection methods to see how well they identify communities explained by feature similarities.

In what follows, after summarising the relevant literature, we introduce our method and demonstrate its performance on synthetic featured networks, for which we control the structural and feature communities as well as the correlations between the two. As a result, we will show that our embeddings, when relying on shared information, outperform the corresponding reference without shared information, and that this performance gap increases with the correlation between network and feature structure since the method can capture the increased joint information. Next, we extensively explore the behaviour of our model on link prediction and node classification on standard benchmark datasets, comparing it to well-known embedding methods. Finally, we close our paper with a short summary and a discussion about potential future directions for our method.

Related work

The advent of increasing computational power coupled with the continuous release and ubiquity of large graph-structured datasets has triggered a surge of research in the field of network embeddings. The main motivation behind this trend is to be able to convert a graph into a low-dimensional space where its structural information and properties are maximally preserved (Cai et al. 2018). The aim is to extract unseen or hard to obtain properties of the network, either directly or by feeding the learned representations to a downstream inference pipeline.

Graph embedding survey: from matrix factorisation to deep learning

In early work, low-dimensional node embeddings were learned for graphs constructed from non-relational data by relying on matrix factorisation techniques. By assuming that the input data lies on a low dimensional manifold, such methods sought to reduce the dimensionality of the data while preserving its structure, and did so by factorising

graph Laplacian eigenmaps (Tenenbaum et al. 2000) or node proximity matrices (Cao et al. 2015).

More recent work has attempted to develop embedding architectures that can use deep learning techniques to compute node representations. DeepWalk (Perozzi et al. 2014), for instance, computes node co-occurrence statistics by sampling the input graph via truncated random walks, and adopts a SkipGram neural language model to maximise the probability of observing the neighbourhood of a node given its embedding. By doing so the learned embedding space preserves second order proximity in the original graph. However, this technique and the ones that followed (Grover and Leskovec 2016; Li et al. 2017) present generalisation caveats, as unobserved nodes during training cannot be meaningfully embedded in the representation space, and the embedding space itself cannot be generalised between graphs. Instead of relying on random walk-based sampling of graphs to feed deep learning architectures, other approaches have used the whole network as input to autoencoders in order to learn, at the bottleneck layer, an efficient representation able to recover proximity information (Wang et al. 2016; Cao et al. 2016; Tran 2018). However, the techniques developed herein remained limited due to the fact that successful deep learning models such as convolutional neural networks require an underlying euclidean structure in order to be applicable.

Geometric deep learning survey: defining convolutional layers on non-euclidean domains

This restriction has been gradually overcome by the development of graph convolutions or Graph Convolutional Networks (GCN). By relying on the definition of convolutions in the spectral domain, Bruna et al. (2013) defined spectral convolution layers based on the spectrum of the graph Laplacian. Several modifications and additions followed and were progressively added to ensure the feasibility of learning on large networks, as well as the spatial localisation of the learned filters (Bronstein et al. 2017; Ying et al. 2018). A key step is made by (Defferrard et al. 2016) with the use of Chebychev polynomials of the Laplacian, in order to avoid having to work in the spectral domain. These polynomials, of order up to r, generate localised filters that behave as a diffusion operator limited to r hops around each vertex. This construction is then further simplified by Kipf and Welling by assuming among others that $r \approx 2$ (Kipf and Welling 2016a).

Recently, these approaches have been extended into more flexible and scalable frameworks. For instance, Hamilton et al. (Hamilton et al. 2017) extended the original GCN framework by enabling the inductive embedding of individual nodes, training a set of functions that learn to aggregate feature information from a node's local neighborhood. In doing so, every node defines a computational graph whose parameters are shared for all the graphs nodes.

More broadly, the combination of GCN with autoencoder architectures has proved fertile for creating new embedding methods. The introduction of probabilistic node embeddings, for instance, has appeared naturally from the application of variational autoencoders to graph data (Rezende et al. 2014; Kingma and Welling 2013; Kipf and Welling 2016b), and has since led to explorations of the uncertainty of embeddings (Bojchevski and Günnemann 2017; Zhu et al. 2018), of appropriate levels of disentanglement and overlap (Mathieu et al. 2018), and of better representation spaces for measuring pairwise embedding distances (see in particular recent applications of the Wasserstein distance between probabilistic embeddings, Zhu et al. 2018;Muzellec and Cuturi 2018).

Such models consistently outperform earlier techniques on different benchmarks and have opened several interesting lines of research in fields ranging from drug design (Duvenaud et al. 2015) to particle physics (Kipf et al. 2018). Most of the more recent approaches mentioned above can incorporate node features (either because they rely on them centrally, or as an add-on). However, with the exception of DANE (Gao and Huang 2018), they mostly do so by assuming that node features are an additional source of information, which is congruent with the network structure (e.g. multi-task learning with shared weights, Tran 2018), or fusing both information types together (Shen et al. 2018). That assumption may not hold in many complex datasets, and it seems important to explore what type of embeddings can be constructed when we lift it, considering different levels of congruence between a network and the features of its nodes.

We therefore set out to make a change to the initial GCN-VAE in order to: (i) create embeddings that are explicitly trained to encode both node features and network structure; (ii) make it so that these embeddings can separate the information that is shared between network and features, from the (possibly non-congruent) information that is specific to either network or features; and (iii) be able to tune the importance that is given to each type of information in the embeddings.

Methods

In this section we present the architecture of the neural network model we use to generate shared feature-structure node embeddings.[1] We take a featured network as input, with structure represented as an adjacency matrix and node features represented as vectors (see below for a formal definition). Our starting point is a GCN-VAE, and our first goal is a multitask reconstruction of both node features and network adjacency matrix. Then, as a second goal, we tune the architecture to be able to scale the number of embedding dimensions dedicated to feature-only reconstruction, adjacency-only reconstruction, or shared feature-adjacency information, while keeping the number of trainable parameters in the model constant.

Multitask graph convolutional autoencoder

We begin with the graph-convolutional variational autoencoder developed by (Kipf and Welling 2016b), which stacks graph-convolutional (GC) layers (Kipf and Welling 2016a) in the encoder part of a variational autoencoder (Rezende et al. 2014; Kingma and Welling 2013) to obtain a lower dimensional embedding of the input structure. This embedding is then used for the reconstruction of the original graph (and in our case, also of the features) in the decoding part of the model. Similarly to Kipf and Welling (2016a), we use two GC layers in our encoder and generate Gaussian-distributed node embeddings at the bottleneck layer of the autoencoder. We now introduce each phase of our embedding method in formal terms.

Encoder

We are given an undirected unweighted featured graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $N = |\mathcal{V}|$ nodes, each node having a D-dimensional feature vector. Loosely following the notations of Kipf and Welling (2016b), we note $\mathbf{A}$ the graph's $N \times N$ adjacency matrix (diagonal elements set to 0), $\mathbf{X}$ the $N \times D$ matrix of node features, and $\mathbf{X}_i$ the D-dimensional feature vector of a node i.

---

[1]The implementation of our model is available online at github.com/ixxi-dante/an2vec.

The encoder part of our model is where F-dimensional node embeddings are generated. It computes $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, two $N \times F$ matrices, which parametrise a stochastic embedding of each node:

$$\boldsymbol{\mu} = \text{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A}) \quad \text{and} \quad \log \boldsymbol{\sigma} = \text{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A}).$$

Here we use two graph-convolutional layers for each parameter set, with shared weights at the first layer and parameter-specific weights at the second layer:

$$\text{GCN}_{\alpha}(\mathbf{X}, \mathbf{A}) = \hat{\mathbf{A}}\text{ReLU}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}_0^{enc})\mathbf{W}_{1,\alpha}^{enc}$$

In this equation, W_0^{enc} and $W_{1,\alpha}^{enc}$ are the weight matrices for the linear transformations of each layer's input; ReLU refers to a rectified linear unit (Nair and Hinton 2010); and following the formalism introduced in Kipf and Welling (2016a), $\hat{\mathbf{A}}$ is the standard normalised adjacency matrix with added self-connections, defined as:

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$$

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$$

$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

Embedding

The parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ produced by the encoder define the distribution of an F-dimensional stochastic embedding $\boldsymbol{\xi}_i$ for each node i, defined as:

$$\boldsymbol{\xi}_i | \mathbf{A}, \mathbf{X} \sim \mathcal{N}\left(\boldsymbol{\mu}_i, \text{diag}\left(\boldsymbol{\sigma}_i^2\right)\right).$$

Thus, for all the nodes we can write a probability density function over a given set of embeddings $\boldsymbol{\xi}$, in the form of an $N \times F$ matrix:

$$q(\boldsymbol{\xi}|\mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\boldsymbol{\xi}_i | \mathbf{A}, \mathbf{X}).$$

Decoder

The decoder part of our model aims to reconstruct both the input node features and the input adjacency matrix by producing parameters of a generative model for each of the inputs. On one hand, the adjacency matrix $\mathbf{A}$ is modelled as a set of independent Bernoulli random variables, whose parameters come from a bilinear form applied to the output of a single dense layer:

$$A_{ij}|\boldsymbol{\xi}_i, \boldsymbol{\xi}_j \sim \text{Ber}(\text{MLB}(\boldsymbol{\xi})_{ij})$$

$$\text{MLB}(\boldsymbol{\xi}) = \text{sigmoid}(\boldsymbol{\gamma}^T \mathbf{W}_{\mathbf{A},1}^{dec} \boldsymbol{\gamma})$$

$$\boldsymbol{\gamma} = \text{ReLU}\left(\boldsymbol{\xi}\mathbf{W}_{\mathbf{A},0}^{dec}\right).$$

Similarly to above, $W_{\mathbf{A},0}^{dec}$ is the weight matrix for the first adjacency matrix decoder layer, and $W_{\mathbf{A},1}^{dec}$ is the weight matrix for the bilinear form which follows.

On the other hand, features can be modelled in a variety of ways, depending on whether they are binary or continuous, and if their norm is constrained or not. Features in our experiments are one-hot encodings, so we model the reconstruction of the feature matrix $\mathbf{X}$ by using N single-draw D-categories multinomial random variables. The parameters

of those multinomial variables are computed from the embeddings with a two-layer perceptron:[2]

$$\mathbf{X}_i|\boldsymbol{\xi}_i \sim \text{Multinomial}(1, \text{MLP}(\boldsymbol{\xi})_i)$$

$$\text{MLP}(\boldsymbol{\xi}) = \text{softmax}\left(\text{ReLU}\left(\boldsymbol{\xi}\mathbf{W}_{\mathbf{X},0}^{dec}\right)\mathbf{W}_{\mathbf{X},1}^{dec}\right)$$

In the above equations, $\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$ refers to the logistic function applied element-wise on vectors or matrices, and $\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$ refers to the normalised exponential function, also applied element-wise, with j running along the rows of matrices (and along the indices of vectors).

Thus we can write the probability density for a given reconstruction as:

$$p(\mathbf{X}, \mathbf{A}|\boldsymbol{\xi}) = p(\mathbf{A}|\boldsymbol{\xi})p(\mathbf{X}|\boldsymbol{\xi})$$

$$p(\mathbf{A}|\boldsymbol{\xi}) = \prod_{i,j=1}^{N} \text{MLB}(\boldsymbol{\xi})_{ij}^{A_{ij}}(1 - \text{MLB}(\boldsymbol{\xi})_{ij})^{1-A_{ij}}$$

$$p(\mathbf{X}|\boldsymbol{\xi}) = \prod_{i=1}^{N}\prod_{j=1}^{D} \text{MLP}(\boldsymbol{\xi})_{ij}^{X_{ij}}$$

Learning

The variational autoencoder is trained by minimising an upper bound to the marginal likelihood-based loss (Rezende et al. 2014) defined as:

$$-\log p(\mathbf{A}, \mathbf{X}) \leq \mathcal{L}(\mathbf{A}, \mathbf{X})$$

$$= D_{KL}(q(\boldsymbol{\xi}|\mathbf{A}, \mathbf{X})||\mathcal{N}(0, \mathbf{I}_F))$$

$$- \mathbb{E}_{q(\boldsymbol{\xi}|\mathbf{A},\mathbf{X})}[\log(p(\mathbf{A}, \mathbf{X}|\boldsymbol{\xi}, \boldsymbol{\theta})p(\boldsymbol{\theta}))]$$

$$= \mathcal{L}_{KL} + \mathcal{L}_{\mathbf{A}} + \mathcal{L}_{\mathbf{X}} + \mathcal{L}_{\boldsymbol{\theta}}$$

Here $\mathcal{L}_{KL}$ is the Kullback-Leibler divergence between the distribution of the embeddings and a Gaussian prior, and $\boldsymbol{\theta}$ is the vector of decoder parameters whose associated loss $\mathcal{L}_{\boldsymbol{\theta}}$ acts as a regulariser for the decoder layers.[3] Computing the adjacency and feature reconstruction losses by using their exact formulas is computationally not tractable, and the standard practice is instead to estimate those losses by using an empirical mean. We generate K samples of the embeddings by using the distribution $q(\boldsymbol{\xi}|\mathbf{A}, \mathbf{X})$ given by the encoder, and average the losses of each of those samples[4] (Rezende et al. 2014; Kingma and Welling 2013):

$$\mathcal{L}_{\mathbf{A}} = -\mathbb{E}_{q(\boldsymbol{\xi}|\mathbf{A},\mathbf{X})}[\log p(\mathbf{A}|\boldsymbol{\xi}, \boldsymbol{\theta})]$$

$$\simeq -\frac{1}{K}\sum_{k=1}^{K}\sum_{i,j=1}^{N}\left[A_{ij}\log\left(\text{MLB}\left(\boldsymbol{\xi}^{(k)}\right)_{ij}\right)\right.$$

$$\left. + (1 - A_{ij})\log\left(1 - \text{MLB}\left(\boldsymbol{\xi}^{(k)}\right)_{ij}\right)\right]$$

---

[2]Other types of node features are modelled according to their constraints and domain. Binary features are modelled as independent Bernoulli random variables. Continuous-range features are modelled as Gaussian random variables in a similar way to the embeddings themselves.

[3]Indeed, following (Rezende et al. 2014) we assume $\boldsymbol{\theta} \sim \mathcal{N}(0, \kappa_{\boldsymbol{\theta}}\mathbf{I})$, such that $\mathcal{L}_{\boldsymbol{\theta}} = -\log p(\boldsymbol{\theta}) = \frac{1}{2}\dim(\boldsymbol{\theta})\log(2\pi\kappa_{\boldsymbol{\theta}}) + \frac{1}{2\kappa_{\boldsymbol{\theta}}}||\boldsymbol{\theta}||_2^2$.

[4]In practice, $K = 1$ is often enough.

$$\mathcal{L}_{\mathbf{X}} = -\mathbb{E}_{q(\boldsymbol{\xi}|\mathbf{A},\mathbf{X})}\big[\log p(\mathbf{X}|\boldsymbol{\xi},\boldsymbol{\theta})\big]$$

$$\simeq -\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{D} X_{ij}\log\left(\mathrm{MLP}\left(\boldsymbol{\xi}^{(k)}\right)_{ij}\right)$$

Finally, for diagonal Gaussian embeddings such as the ones we use, $\mathcal{L}_{KL}$ can be expressed directly (Kingma and Welling 2013):

$$\mathcal{L}_{KL} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{F} \mu_{ij}^{2} + \sigma_{ij}^{2} - 2\log\sigma_{ij} - 1$$

Loss adjustments

In practice, to obtain useful results a few adjustments are necessary to this loss function. First, given the high sparsity of real-world graphs, the A_{ij} and $1-A_{ij}$ terms in the adjacency loss must be scaled respectively up and down in order to avoid globally near-zero link reconstruction probabilities. Instead of penalising reconstruction proportionally to the overall number of errors in edge prediction, we want false negatives (A_{ij} terms) and false positives ($1 - A_{ij}$ terms) to contribute equally to the reconstruction loss, independent of graph sparsity. Formally, let $d = \frac{\sum_{ij} A_{ij}}{N^2}$ denote the density of the graph's adjacency matrix ($d = \frac{N-1}{N} \times \mathrm{density}(\mathcal{G})$); then we replace $\mathcal{L}_{\mathbf{A}}$ by the following re-scaled estimated loss (the so-called "balanced cross-entropy"):

$$\tilde{\mathcal{L}}_{\mathbf{A}} = -\frac{1}{K}\sum_{k=1}^{K}\sum_{i,j=1}^{N}\frac{1}{2}\Bigg[\frac{A_{ij}}{d}\log\left(\mathrm{MLB}\left(\boldsymbol{\xi}^{(k)}\right)_{ij}\right)$$

$$+ \frac{1-A_{ij}}{1-d}\log\left(1 - \mathrm{MLB}\left(\boldsymbol{\xi}^{(k)}\right)_{ij}\right)\Bigg]$$

Second, we correct each component loss for its change of scale when the shapes of the inputs and the model parameters change: $\mathcal{L}_{KL}$ is linear in N and F, $\tilde{\mathcal{L}}_{\mathbf{A}}$ is quadratic in N, and $\mathcal{L}_{\mathbf{X}}$ is linear in N (but not in F, remember that $\sum_j X_{ij} = 1$ since each $\mathbf{X}_i$ is a single-draw multinomial).

Beyond dimension scaling, we also wish to keep the values of $\tilde{\mathcal{L}}_{\mathbf{A}}$ and $\mathcal{L}_{\mathbf{X}}$ comparable and, doing so, maintain a certain balance between the difficulty of each task. As a first approximation to the solution, and in order to avoid more elaborate schemes which would increase the complexity of our architecture (such as Chen et al. 2018), we divide both loss components by their values at maximum uncertainty,[5] respectively $\log 2$ and $\log D$.

Finally, we make sure that the regulariser terms in the loss do not overpower the actual learning terms (which are now down-scaled close to 1) by adjusting $\kappa_{\boldsymbol{\theta}}$ and an additional factor, κ_{KL}, which scales the Kullback-Leibler term.[6] These adjustments lead us to the final total loss the model is trained for:

$$\mathcal{L} = \frac{\tilde{\mathcal{L}}_{\mathbf{A}}}{N^2\log 2} + \frac{\mathcal{L}_{\mathbf{X}}}{N\log D} + \frac{\mathcal{L}_{KL}}{NF\kappa_{KL}} + \frac{||\boldsymbol{\theta}||_2^2}{2\kappa_{\boldsymbol{\theta}}}$$

where we have removed constant terms with respect to trainable model parameters.

Scaling shared information allocation

The model we just presented uses all dimensions of the embeddings indiscriminately to reconstruct the adjacency matrix and the node features. While this can

---

[5]That is, $p(A_{ij}|\boldsymbol{\xi},\boldsymbol{\theta}) = \frac{1}{2}$ $\forall i,j$, and $p(X_{ij}|\boldsymbol{\xi},\boldsymbol{\theta}) = \frac{1}{D}$ $\forall i,j$.
[6]We use $\kappa_{KL} = 2\kappa_{\boldsymbol{\theta}} = 10^3$.

be useful in some cases, it cannot adapt to different interdependencies between graph structure and node features; in cases where the two are not strongly correlated, the embeddings would lose information by conflating features and graph structure. Therefore our second aim is to adjust the dimensions of the embeddings used exclusively for feature reconstruction, or for adjacency reconstruction, or used for both.

In a first step, we restrict which part of a node's embedding is used for each task. Let $F_{\mathbf{A}}$ be the number of embedding dimensions we will allocate to adjacency matrix reconstruction only, $F_{\mathbf{X}}$ the number of dimensions allocated to feature reconstruction only, and $F_{\mathbf{AX}}$ the number of dimensions allocated to both. We have $F_{\mathbf{A}} + F_{\mathbf{AX}} + F_{\mathbf{X}} = F$. We further introduce the following notation for the restriction of the embedding of node i to a set of dedicated dimensions $\{a, \dots, b\}$:[7]

$$\boldsymbol{\xi}_{i,a:b} = (\xi_{ij})_{j \in \{a,\dots,b\}}$$

This extends to the full matrix of embeddings similarly:

$$\boldsymbol{\xi}_{a:b} = (\xi_{ij})_{i \in \{1,\dots,N\}, j \in \{a,\dots,b\}}$$

Using these notations we adapt the decoder to reconstruct adjacency and features as follows:

$$A_{ij} | \boldsymbol{\xi}_{i,1:F_{\mathbf{A}}+F_{\mathbf{AX}}}, \boldsymbol{\xi}_{j,1:F_{\mathbf{A}}+F_{\mathbf{AX}}} \sim \mathrm{Ber}(\mathrm{MLB}(\boldsymbol{\xi}_{1:F_{\mathbf{A}}+F_{\mathbf{AX}}})_{ij})$$

$$\mathbf{X}_i | \boldsymbol{\xi}_{i,F_{\mathbf{A}}+1:F} \sim \mathrm{Multinomial}(1, \mathrm{MLP}(\boldsymbol{\xi}_{F_{\mathbf{A}}+1:F})_i)$$

In other words, adjacency matrix reconstruction relies on $F_{\mathbf{A}} + F_{\mathbf{AX}}$ embedding dimensions, feature reconstruction relies on $F_{\mathbf{X}} + F_{\mathbf{AX}}$ dimensions, and $F_{\mathbf{AX}}$ overlapping dimensions are shared between the two. Our reasoning is that for datasets where the dependency between features and network structure is strong, shallow models with higher overlap value will perform better than models with the same total embedding dimensions F and less overlap, or will perform on par with models that have more total embedding dimensions and less overlap. Indeed, the overlapping model should be able to extract the information shared between features and network structure and store it in the overlapping dimensions, while keeping the feature-specific and structure-specific information in their respective embedding dimensions. This is to compare to the non-overlapping case, where shared network-feature information is stored redundantly, both in feature- and structure-specific embeddings, at the expense of a larger number of distinct dimensions.

Therefore, to evaluate the performance gains of this architecture, one of our measures is to compare the final loss for different hyperparameter sets, keeping $F_{\mathbf{A}} + F_{\mathbf{AX}}$ and $F_{\mathbf{X}} + F_{\mathbf{AX}}$ fixed and varying the overlap size $F_{\mathbf{AX}}$. Now, to make sure the training losses for different hyperparameter sets are comparable, we must maintain the overall number of trainable parameters in the model fixed. The decoder already has a constant number of trainable parameters, since it only depends on the number of dimensions used for decoding features ($F_{\mathbf{X}} + F_{\mathbf{AX}}$) and adjacency matrix ($F_{\mathbf{A}} + F_{\mathbf{AX}}$), which are themselves fixed.

---

[7]Note that the order of the indices does not change the training results, as the model has no notion of ordering inside its layers. What follows is valid for any permutation of the dimensions, and the actual indices only matter to downstream interpretation of the embeddings after training.

On the other hand, the encoder requires an additional change. We maintain the dimensions of the encoder-generated μ and σ parameters fixed at $F_A+2F_{AX}+F_X$ (independently from F_{AX}, given the constraints above), and reduce those outputs to $F_A+F_{AX}+F_X$ dimensions by averaging dimensions $\{F_A+1,\ldots,F_A+F_{AX}\}$ and $\{F_A+F_{AX}+1,\ldots,F_A+2F_{AX}\}$ together.[8] In turn, this model maintains a constant number of trainable parameters, while allowing us to adjust the number of dimensions F_{AX} shared by feature and adjacency reconstruction (keeping F_A+F_{AX} and F_X+F_{AX} constant). Figure 1 schematically represents this architecture.

Results

We are interested in measuring two main effects: first, the variation in model performance as we increase the overlap in the embeddings, and second, the capacity of the embeddings with overlap (versus no overlap) to capture and benefit from dependencies between graph structure and node features. To that end, we train overlapping and non-overlapping models on synthetic data with different degrees of correlation between network structure and node features.

Synthetic featured networks

We use a Stochastic Block Model (Holland et al. 1983) to generate synthetic featured networks, each with M communities of $n = 10$ nodes, with intra-cluster connection probabilities of 0.25, and with inter-cluster connection probabilities of 0.01. Each node is initially assigned a colour which encodes its feature community; we shuffle the colours of a fraction $1-\alpha$ of the nodes, randomly sampled. This procedure maintains constant the overall count of each colour, and lets us control the correlation between the graph structure and node features by moving α from 0 (no correlation) to 1 (full correlation).

Node features are represented by a one-hot encoding of their colour (therefore, in all our scenarios, the node features have dimension $M = N/n$). However, since in this case all the nodes inside a community have exactly the same feature value, the model can have difficulties differentiating nodes from one another. We therefore add a small Gaussian noise ($\sigma = .1$) to make sure that nodes in the same community can be distinguished from one another.

Note that the feature matrix has less degrees of freedom than the adjacency matrix in this setup, a fact that will be reflected in the plots below. However, opting for this minimal generative model lets us avoid the parameter exploration of more complex schemes for feature generation, while still demonstrating the effectiveness of our model.
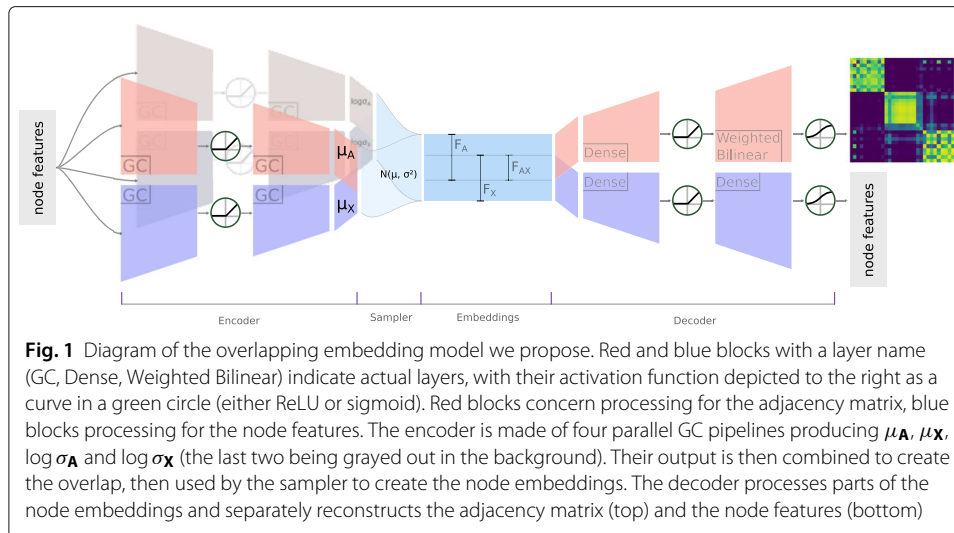
Comparison setup

To evaluate the efficiency of our model in terms of capturing meaningful correlations between network and features, we compare overlapping and non-overlapping models as

---

[8]Formally:

$$\tilde{\mu} = \mu_{1:F_A} \| \frac{1}{2}\left(\mu_{F_A+1:F_A+F_{AX}} + \mu_{F_A+F_{AX}+1:F_A+2F_{AX}}\right)$$

$$\| \mu_{F_A+2F_{AX}+1:F_A+2F_{AX}+F_X}$$

$$\log\tilde{\sigma} = \log\sigma_{1:F_A} \| \frac{1}{2}\left(\log\sigma_{F_A+1:F_A+F_{AX}} + \log\sigma_{F_A+F_{AX}+1:F_A+2F_{AX}}\right)$$

$$\| \log\sigma_{F_A+2F_{AX}+1:F_A+2F_{AX}+F_X}$$

where $\|$ denotes concatenation along the columns of the matrices.

Fig. 1 Diagram of the overlapping embedding model we propose. Red and blue blocks with a layer name (GC, Dense, Weighted Bilinear) indicate actual layers, with their activation function depicted to the right as a curve in a green circle (either ReLU or sigmoid). Red blocks concern processing for the adjacency matrix, blue blocks processing for the node features. The encoder is made of four parallel GC pipelines producing $\mu_{\mathbf{A}}$, $\mu_{\mathbf{X}}$, $\log \sigma_{\mathbf{A}}$ and $\log \sigma_{\mathbf{X}}$ (the last two being grayed out in the background). Their output is then combined to create the overlap, then used by the sampler to create the node embeddings. The decoder processes parts of the node embeddings and separately reconstructs the adjacency matrix (top) and the node features (bottom)

follows. For a given maximum number of embedding dimensions F_{max}, the overlapping models keep constant the number of dimensions used for adjacency matrix reconstruction and the number of dimensions used for feature reconstruction, with the same amount allocated to each task: $F_{\mathbf{A}}^{ov} + F_{\mathbf{AX}}^{ov} = F_{\mathbf{X}}^{ov} + F_{\mathbf{AX}}^{ov} = \frac{1}{2}F_{max}$. However they vary the overlap $F_{\mathbf{AX}}^{ov}$ from 0 to $\frac{1}{2}F_{max}$ by steps of 2. Thus the total number of embedding dimensions F varies from F_{max} to $\frac{1}{2}F_{max}$, and as F decreases, $F_{\mathbf{AX}}^{ov}$ increases. We call one such model $\mathcal{M}_F^{ov}$.

Now for a given overlapping model $\mathcal{M}_F^{ov}$, we define a reference model $\mathcal{M}_F^{ref}$, which has the same total number of embedding dimensions, but without overlap: $F_{\mathbf{AX}}^{ref} = 0$, and $F_{\mathbf{A}}^{ref} = F_{\mathbf{X}}^{ref} = \frac{1}{2}F$ (explaining why we vary F with steps of 2). Note that while the reference model has the same information bottleneck as the overlapping model, it has less trainable parameters in the decoder, since $F_{\mathbf{A}}^{ref} + F_{\mathbf{AX}}^{ref} = F_{\mathbf{X}}^{ref} + F_{\mathbf{AX}}^{ref} = \frac{1}{2}F$ will decrease as F decreases. Nevertheless, this will not be a problem for our measures, since we will be mainly looking at the behaviour of a given model for different values of α (i.e. the feature-network correlation parameter).
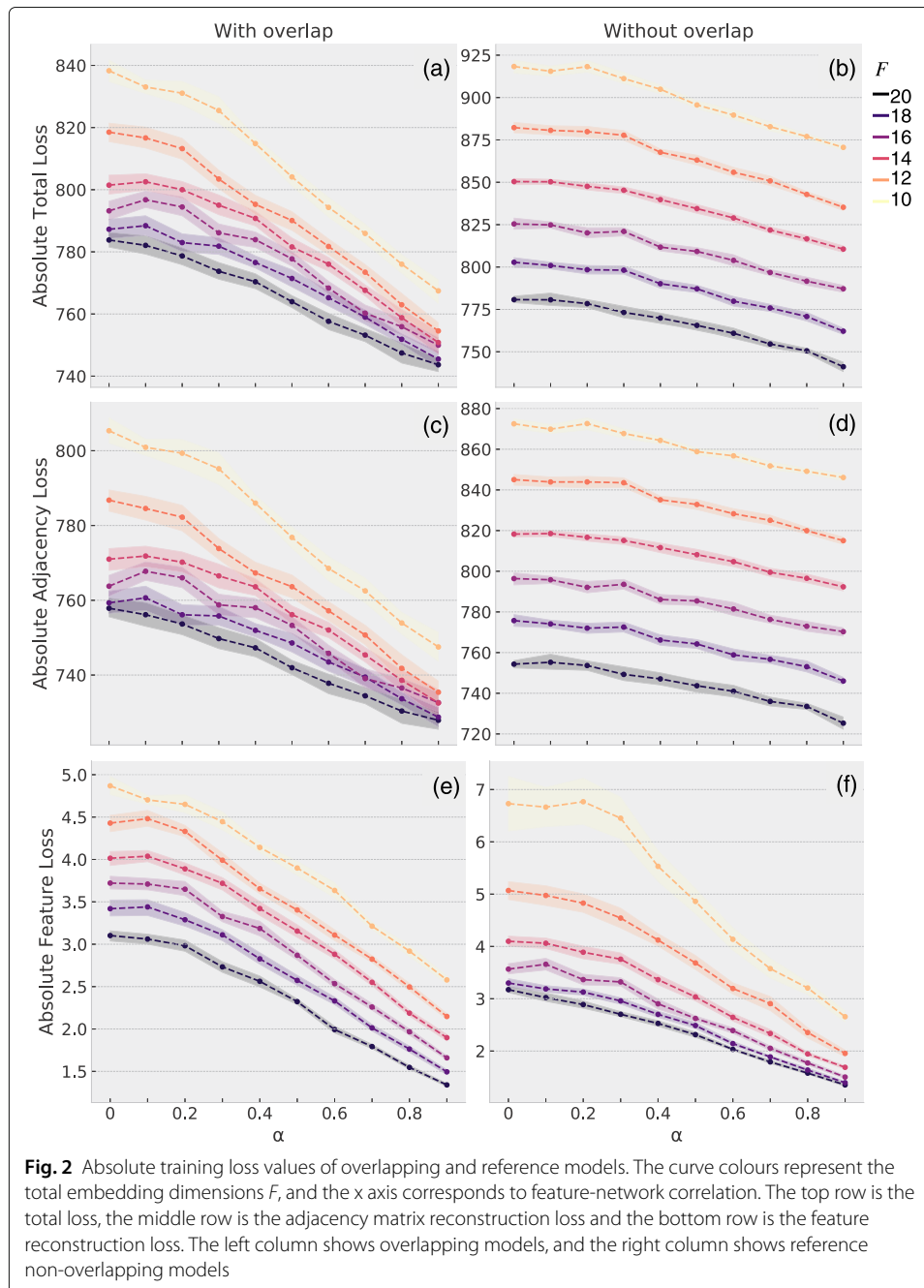
For our calculations (if not noted otherwise) we use synthetic networks of $N = 1000$ nodes (i.e. 100 clusters), and set the maximum embedding dimensions F_{max} to 20. For all models, we set the intermediate layer in the encoder and the two intermediate layers in the decoder to an output dimension of 50, and the internal number of samples for loss estimation at $K = 5$. We train our models for 1000 epochs using the Adam optimiser (Kingma and Ba 2014) with a learning rate of 0.01 (following Kipf and Welling 2016b), after initialising weights following Glorot and Bengio (2010). For each combination of F and α, the training of the overlapping and reference models is repeated 20 times on independent featured networks.

Since the size of our synthetic data is constant, and we average training results over independently sampled data sets, we can meaningfully compare the averaged training losses of models with different parameters. We therefore take the average best training loss of a model to be our main measure, indicating the capacity to reconstruct an input data set for a given information bottleneck and embedding overlap.

Advantages of overlap

Absolute loss values

Figure 2 shows the variation of the best training loss (total loss, adjacency reconstruction loss, and feature reconstruction loss) for both overlapping and reference models, with α ranging from 0 to 1 and F decreasing from 20 to 10 by steps of 2. One curve in these plots represents the variation in losses of a model with fixed F for data sets with increasing correlation between network and features; each point aggregates 20 independent trainings, used to bootstrap 95% confidence intervals.

Fig. 2 Absolute training loss values of overlapping and reference models. The curve colours represent the total embedding dimensions F, and the x axis corresponds to feature-network correlation. The top row is the total loss, the middle row is the adjacency matrix reconstruction loss and the bottom row is the feature reconstruction loss. The left column shows overlapping models, and the right column shows reference non-overlapping models

We first see that all losses, whether for overlapping model or reference, decrease as we move from the uncorrelated scenario to the correlated scenario. This is true despite the fact that the total loss is dominated by the adjacency reconstruction loss, as feature reconstruction is an easier task overall. Second, recall that the decoder in a reference model has less parameters than its corresponding overlapping model of the same F dimensions (except for zero overlap), such that the reference is less powerful and produces higher training losses. The absolute values of the losses for overlap and reference models are therefore not directly comparable. However, the changes in slopes are meaningful. Indeed, we note that the curve slopes are steeper for models with higher overlap (lower F) than for lower overlap (higher F), whereas they seem relatively independent for the reference models of different F. In other words, as we increase the overlap, our models seem to benefit more from an increase in network-feature correlation than what a reference model benefits.
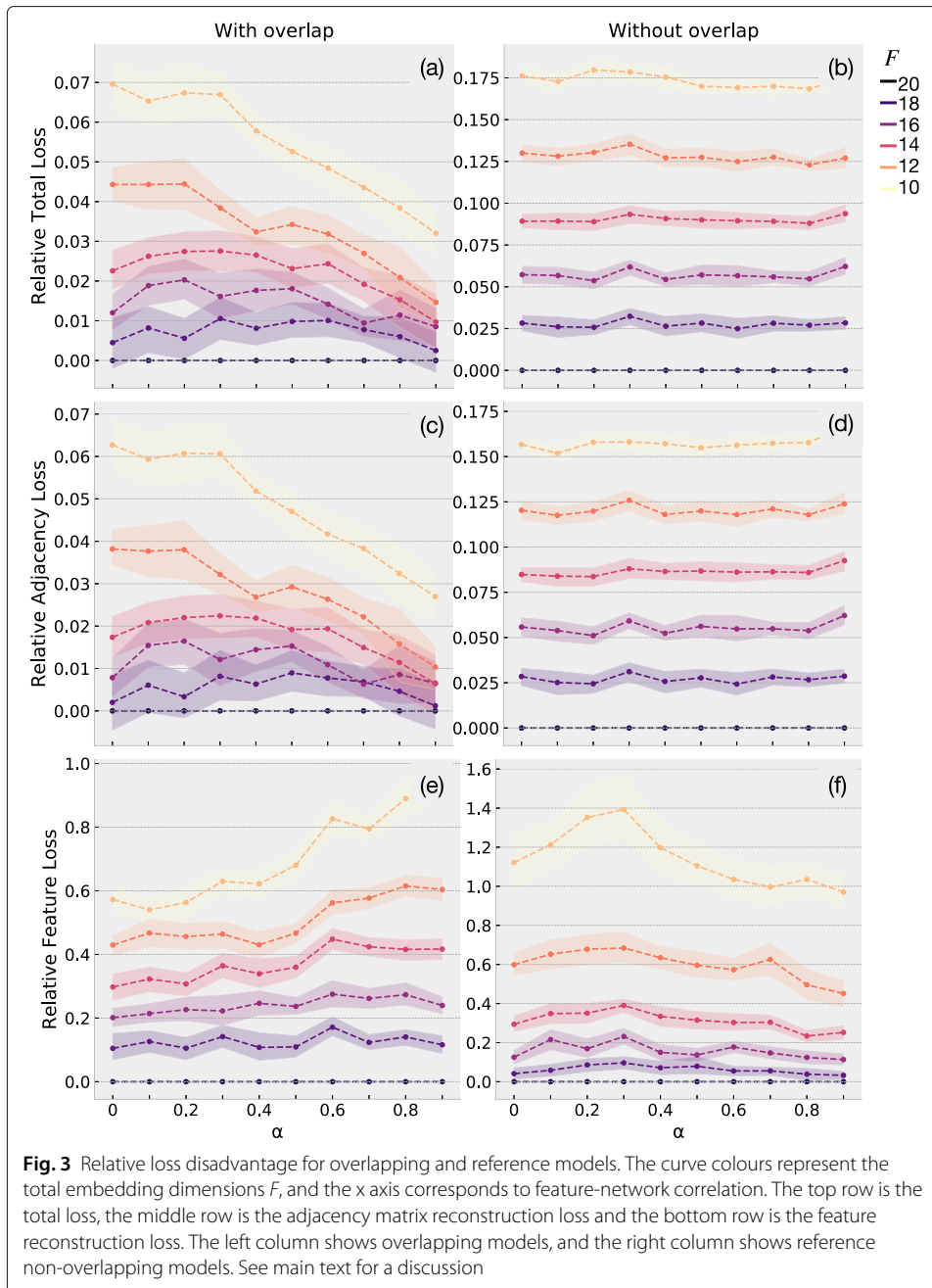
Relative loss disadvantage

In order to assess this trend more reliably, we examine losses relative to the maximum embedding models. Figure 3 plots the loss disadvantage that overlap and reference models have compared to their corresponding model with $F = F_{max}$, that is, $\frac{\mathcal{L}_{\mathcal{M}_F} - \mathcal{L}_{\mathcal{M}_{Fmax}}}{\mathcal{L}_{\mathcal{M}_{Fmax}}}$. We call this the *relative loss disadvantage* of a model. In this plot, the height of a curve thus represents the magnitude of the decrease in performance of a model $\mathcal{M}_F^{ov|ref}$ relative to the model with maximum embedding size, $\mathcal{M}_{Fmax}^{ov|ref}$. Note that for both the overlap model and the reference model, moving along one of the curves does not change the number of trainable parameters in the model.

As the correlation between network and features increases, we see that the relative loss disadvantage decreases in overlap models, and that the effect is stronger for higher overlaps. In other words, when the network and features are correlated, the overlap captures this joint information and compensates for the lower total number of dimensions (compared to $\mathcal{M}_{Fmax}^{ov|ref}$): the model achieves a better performance than when network and features are more independent. Strikingly, for the reference model these curves are flat, thus indicating no variation in relative loss disadvantage with varying network-feature correlations in these cases. This confirms that the new measure successfully controls for the baseline decrease of absolute loss values when the network-features correlation increases, as observed in Fig. 2. Our architecture is therefore capable of capturing and taking advantage of some of the correlation by leveraging the overlap dimensions of the embeddings.

Finally note that for high overlaps, the feature reconstruction loss value actually increases a little when α grows. The behaviour is consistent with the fact that the total loss is dominated by the adjacency matrix loss (the hardest task). In this case it seems that the total loss is improved more by exploiting the gain of optimising for adjacency matrix reconstruction, and paying the small cost of a lesser feature reconstruction, than decreasing both adjacency matrix and feature losses together. If wanted, this strategy could be controlled using a gradient normalisation scheme such as Chen et al. (2018).

Standard benchmarks

Finally we compare the performance of our architecture to other well-known embedding methods, namely spectral clustering (SC) (Tang and Liu 2011), DeepWalk (DW) (Perozzi

Fig. 3 Relative loss disadvantage for overlapping and reference models. The curve colours represent the total embedding dimensions *F*, and the x axis corresponds to feature-network correlation. The top row is the total loss, the middle row is the adjacency matrix reconstruction loss and the bottom row is the feature reconstruction loss. The left column shows overlapping models, and the right column shows reference non-overlapping models. See main text for a discussion

et al. 2014), the vanilla non-variational and variational Graph Autoencoders (GAE and VGAE) (Kipf and Welling 2016b), and GraphSAGE (Hamilton et al. 2017) which we look at in more detail. We do so on two tasks: (i) the link prediction task introduced by Kipf and Welling (2016b) and (ii) a node classification task, both on the Cora, CiteSeer and PubMed datasets, which are regularly used as citation network benchmarks in the literature (Sen et al. 2008; Namata et al. 2012). Note that neither SC nor DW support feature information as an input.

The Cora and CiteSeer datasets are citation networks made of respectively 2708 and 3312 machine learning articles, each assigned to a small number of document classes (7

for Cora, 6 for CiteSeer), with a bag-of-words feature vector for each article (respectively 1433 and 3703 words). The PubMed network is made of 19717 diabetes-related articles from the PubMed database, each assigned to one of three classes, with article feature vectors containing *term frequency-inverse document frequency* (TF/IDF) scores for 500 words.

Link prediction

The link prediction task consists in training a model on a version of the datasets where part of the edges has been removed, while node features are left intact. A test set is formed by randomly sampling 15% of the edges combined with the same number of random disconnected pairs (non-edges). Subsequently the model is trained on the remaining dataset where 15% of the real edges are missing.

We pick hyperparameters such that the restriction of our model to VGAE would match the hyperparameters used by Kipf and Welling (2016b). That is a 32-dimensions intermediate layer in the encoder and the two intermediate layers in the decoder, and 16 embedding dimensions for each reconstruction task ($F_A + F_{AX} = F_X + F_{AX} = 16$). We call the zero-overlap and the full-overlap versions of this model AN2VEC-0 and AN2VEC-16 respectively. In addition, we test a variant of these models with a shallow adjacency matrix decoder, consisting of a direct inner product between node embeddings, while keeping the two dense layers for feature decoding. Formally: $A_{ij}|\boldsymbol{\xi}_i, \boldsymbol{\xi}_j \sim \text{Ber}(\text{sigmoid}(\boldsymbol{\xi}_i^T \boldsymbol{\xi}_j))$. This modified overlapping architecture can be seen as simply adding the feature decoding and embedding overlap mechanics to the vanilla VGAE. Consistently, we call the zero-overlap and full-overlap versions AN2VEC-S-0 and AN2VEC-S-16.

We follow the test procedure laid out by Kipf and Welling (2016b): we train for 200 epochs using the Adam optimiser (Kingma and Ba 2014) with a learning rate of .01, initialise weights following Glorot and Bengio (2010), and repeat each condition 10 times. The $\boldsymbol{\mu}$ parameter of each node's embedding is then used for link prediction (i.e. the parameter is put through the decoder directly without sampling), for which we report *area under the ROC curve* and *average precision* scores in Table 1.[9]

We argue that AN2VEC-0 and AN2VEC-16 should have somewhat poorer performance than VGAE. These models are required to reconstruct an additional output, which is not

Table 1 Link prediction task in citation networks

Method	Cora		CiteSeer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6 ±0.01	88.5 ±0.00	80.5 ±0.01	85.0 ±0.01	84.2 ±0.02	87.8 ±0.01
DW	83.1 ±0.01	85.0 ±0.00	80.5 ±0.02	83.6 ±0.01	84.4 ±0.00	84.1 ±0.00
GAE	91.0 ±0.02	92.0 ±0.03	89.5 ±0.04	89.9 ±0.05	**96.4** ±0.00	**96.5** ±0.00
VGAE	91.4 ±0.01	92.6 ±0.01	90.8 ±0.02	92.0 ±0.02	94.4 ±0.02	94.7 ±0.02
AN2VEC-0	89.5 ±0.01	90.6 ±0.01	91.2 ±0.01	91.5 ±0.02	91.8 ±0.01	93.2 ±0.01
AN2VEC-16	89.4 ±0.01	90.2 ±0.01	91.1 ±0.01	91.3 ±0.02	92.1 ±0.01	92.8 ±0.01
AN2VEC-S-0	92.9 ±0.01	93.4 ±0.01	94.3 ±0.01	94.8 ±0.01	95.1 ±0.01	95.4 ±0.01
AN2VEC-S-16	**93.0** ±0.01	**93.5** ±0.00	**94.9** ±0.00	**95.1** ±0.00	93.1 ±0.01	93.1 ±0.01

SC, DW, GAE and VGAE values are from Kipf and Welling (2016b). Error values indicate the sample standard deviation
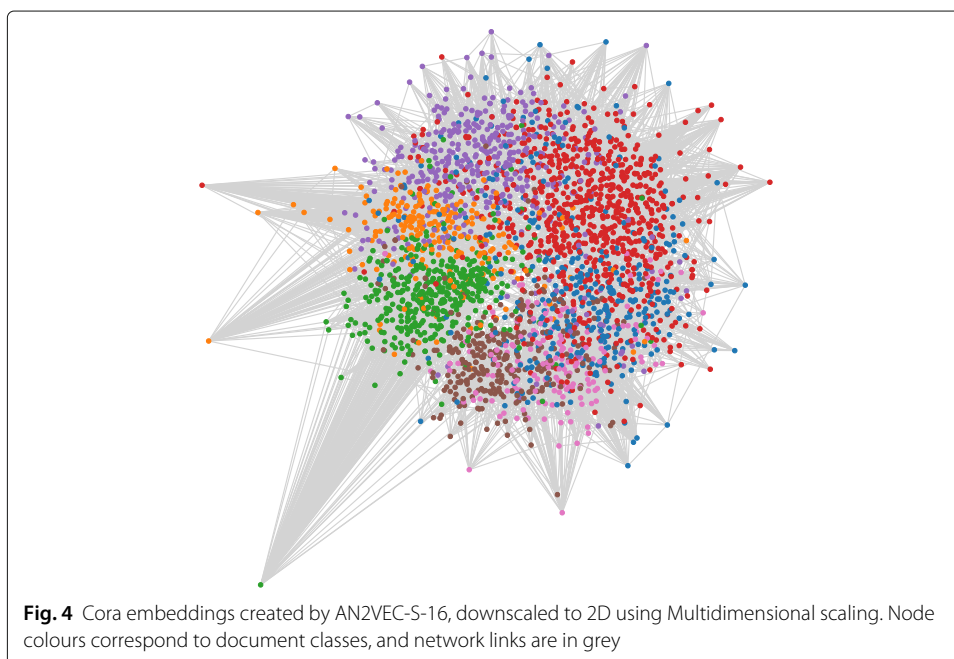
[9]Note that in Kipf and Welling (2016b), the training set is also 85% of the full dataset, and test and validation sets are formed with the remaining edges, respectively 10% and 5% (and the same amount of non-edges). Here, since we use the same hyperparameters as Kipf and Welling (2016b) we do not need a validation set. We therefore chose to use the full 15% remaining edges (with added non-edges) as a test set, as explained above.

directly used for the link prediction task at hand. First results confirmed our intuition. However, we found that the shallow decoder models AN2VEC-S-0 and AN2VEC-S-16 perform consistently better than the vanilla VGAE for Cora and CiteSeer while their deep counterparts (AN2VEC-0 and AN2VEC-16) outperforms VGAE for all datasets. As neither AN2VEC-0 nor AN2VEC-16 exhibited over-fitting, this behaviour is surprising and calls for further explorations which are beyond the scope of this paper (in particular, this may be specific to the link prediction task). Nonetheless, the higher performance of AN2VEC-S-0 and AN2VEC-S-16 over the vanilla VGAE on Cora and CiteSeer confirms that including feature reconstruction in the constraints of node embeddings is capable of increasing link prediction performance when feature and structure are not independent (consistent with Gao and Huang 2018; Shen et al. 2018; Tran 2018). An illustration of the embeddings produced by AN2VEC-S-16 on Cora is shown in Fig. 4.

On the other hand, performance of AN2VEC-S-0 on PubMed is comparable with GAE and VGAE, while AN2VEC-S-16 has slightly lower performance. The fact that lower overlap models perform better on this dataset indicates that features and structure are less congruent here than in Cora or CiteSeer (again consistent with the comparisons found in Tran 2018). Despite this, an advantage of the embeddings produced by the AN2VEC-S-16 model is that they encode *both* the network structure and the node features, and can therefore be used for downstream tasks involving both types of information.

We further explore the behaviour of the model for different sizes of the training set, ranging from 10% to 90% of the edges in each dataset (reducing the training set accordingly), and compare the behaviour of AN2VEC to GraphSAGE. To make the comparison meaningful we train two variants of the two-layer GraphSAGE model with mean aggregators and no bias vectors: one with an intermediate layer of 32 dimensions and an embedding layer of 16 dimensions (roughly equivalent in dimensions to the full overlap AN2VEC models), the second with an intermediate layer of 64 dimensions and an embedding layer of 32 dimensions (roughly equivalent to no overlap in AN2VEC). Both

Fig. 4 Cora embeddings created by AN2VEC-S-16, downscaled to 2D using Multidimensional scaling. Node colours correspond to document classes, and network links are in grey

layers use neighbourhood sampling, 10 neighbours for the first layer and 5 for the second. Similarly to the shallow AN2VEC decoder, each pair of node embeddings is reduced by inner product and a sigmoid activation, yielding a scalar prediction between 0 and 1 for each possible edge. The model is trained on minibatches of 50 edges and non-edges (edges generated with random walks of length 5), learning rate 0.001, and 4 total epochs. Note that on Cora, one epoch represents about 542 minibatches,[10] such that 4 epochs represent about 2166 gradient updates; thus with a learning rate of 0.001, we remain comparable to the 200 full batches with learning rate 0.01 used to train AN2VEC.

Figure 5 plots the AUC produced by AN2VEC and GraphSAGE for different training set sizes and different embedding sizes (and overlaps, for AN2VEC), for each dataset. As expected, the performance of both models decreases as the size of the test set increases, though less so for AN2VEC. For Cora and CiteSeer, similarly to Table 1, higher overlaps and a shallow decoder in AN2VEC give better performance. Notably, the shallow decoder version of AN2VEC with full overlap is still around .75 for a test size of 90%, whereas both GraphSAGE variants are well below .65. For PubMed, as in Table 1, the behaviour is different to the first two datasets, as overlaps 0 and 16 yield the best results. As for Cora and CiteSeer, the approach taken by AN2VEC gives good results: with a test size of 90%, all AN2VEC deep decoder variants are still above .75 (and shallow decoders above .70), whereas both GraphSAGE variants are below .50.

Node classification

Since the embeddings produced also encode feature information, we then evaluate the model's performance on a node classification task. Here the models are trained on a version of the dataset where a portion of the nodes (randomly selected) have been removed; next, a logistic classifier[11] is trained on the embeddings to classify training nodes into their classes; finally, embeddings are produced for the removed nodes, for which we show the F1 scores of the classifier.
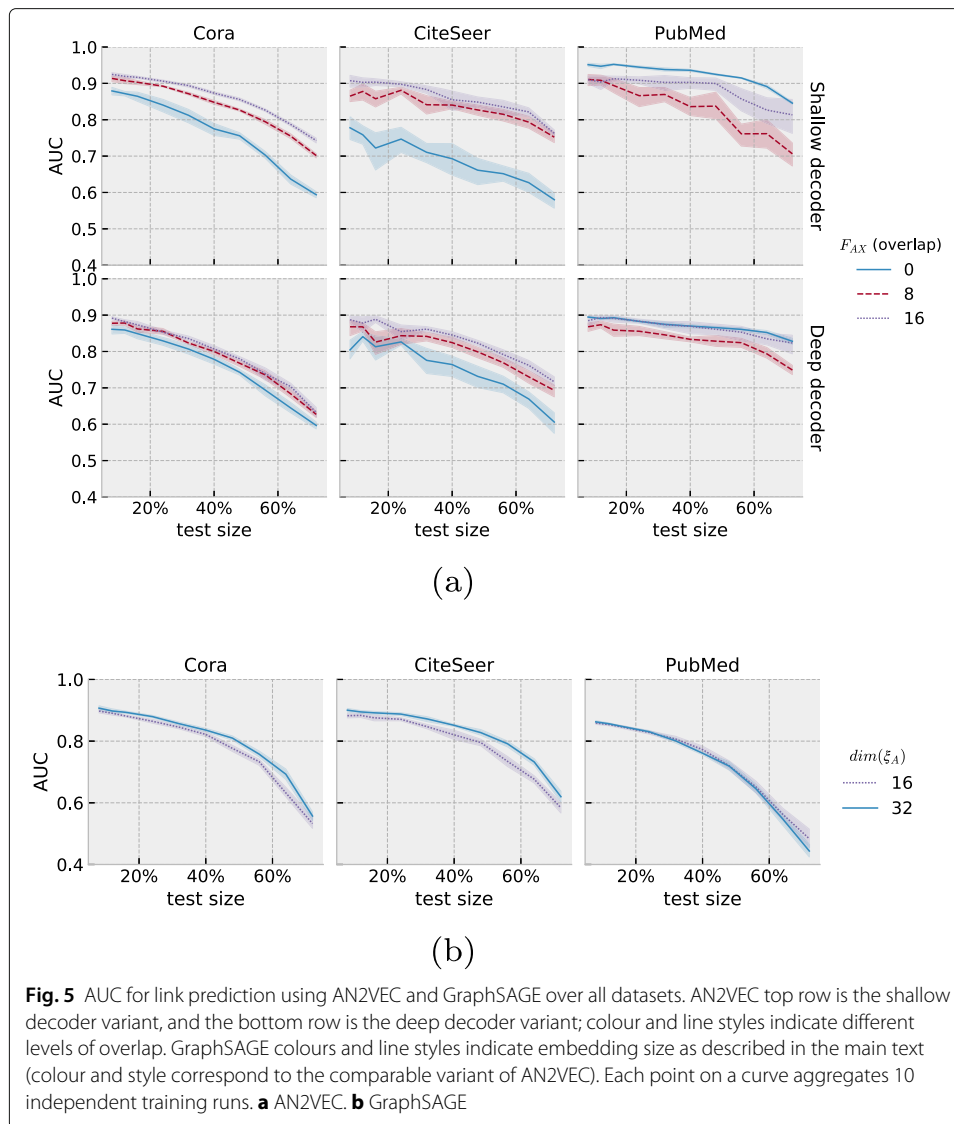
Figure 6 shows the results for AN2VEC and GraphSAGE on all datasets. The scale of the reduction in performance as the test size increases is similar for both models (and similar to the behaviour for link prediction), though overlap and shallow versus deep decoding seem to have less effect. Still, the deep decoder is less affected by the change in test size than the shallow decoder; and contrary to the link prediction case, the 0 overlap models perform best (on all datasets). Overall, the performance levels of GraphSAGE and AN2VEC on this task are quite similar, with slightly better results of AN2VEC on Cora, slightly stronger performance for GraphSAGE on CiteSeer, and mixed behaviour on PubMed (AN2VEC is better for small test sizes and worse for large test sizes).

Variable embedding size

We also explore the behaviour of AN2VEC for different embedding sizes. We train models with $F_\mathbf{A} = F_\mathbf{X} \in \{8, 16, 24, 32\}$ and overlaps 0, 8, 16, 24, 32 (whenever there are enough dimensions to do so), with variable test size. Figure 7 shows the AUC scores for link prediction, and Fig. 8 shows the F1-micro scores for node classification, both on Cite-Seer (the behaviour is similar on Cora, though less salient). For link prediction, beyond
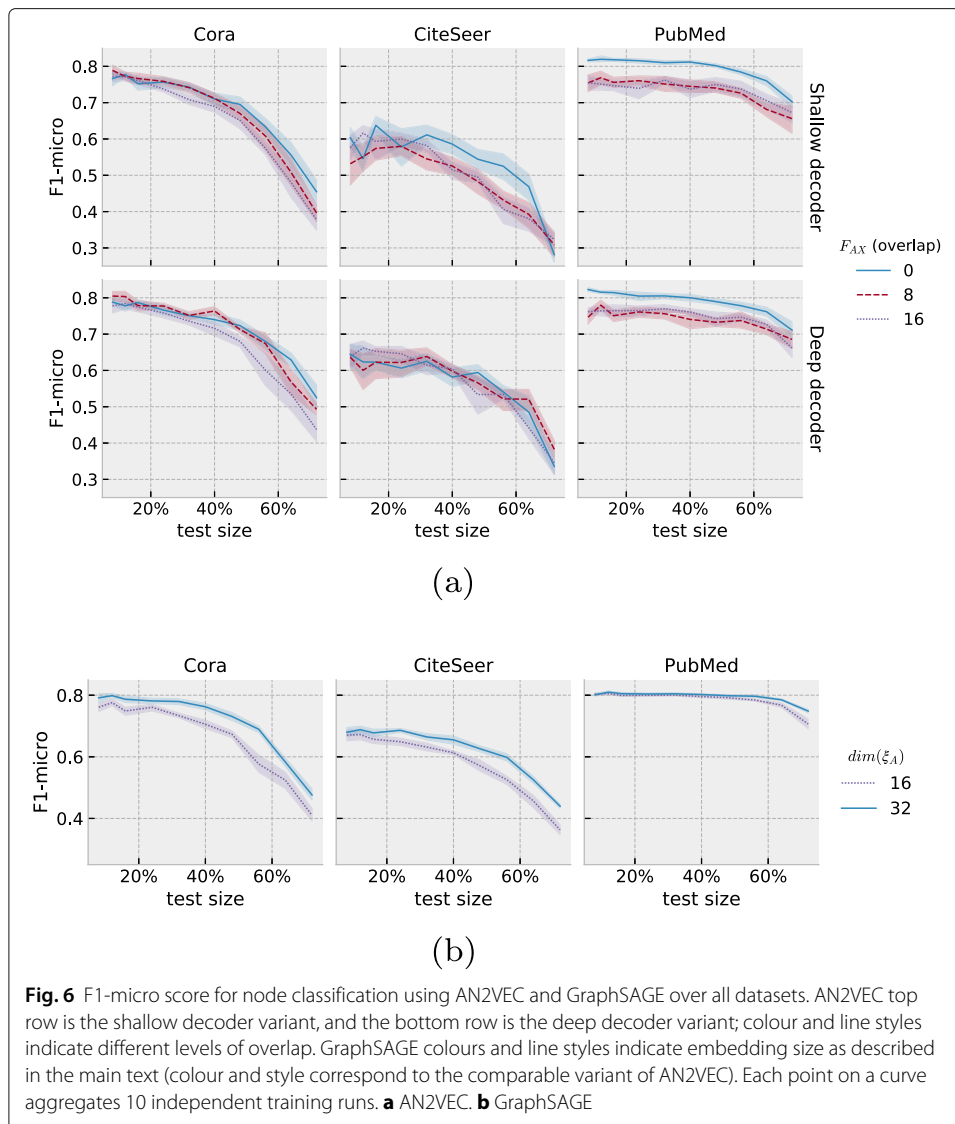
---

[10]One epoch is 2708 nodes × 5 edges per node × 2 (for non-edges) = 27080 training edges or non-edges; divided by 50, this makes 541.6 minibatches per epoch.
[11]Using Scikit-learn's (Pedregosa et al. 2011) interface to the liblinear library, with one-vs-rest classes.

Fig. 5 AUC for link prediction using AN2VEC and GraphSAGE over all datasets. AN2VEC top row is the shallow decoder variant, and the bottom row is the deep decoder variant; colour and line styles indicate different levels of overlap. GraphSAGE colours and line styles indicate embedding size as described in the main text (colour and style correspond to the comparable variant of AN2VEC). Each point on a curve aggregates 10 independent training runs. **a** AN2VEC. **b** GraphSAGE

confirming trends already observed previously, we see that models with less total embedding dimensions perform slightly better than models with more total dimensions. More interestingly, all models seem to reach a plateau at overlap 8, and then exhibit a slightly fluctuating behaviour as overlap continues to increase (in models that have enough dimensions to do so). This is valid for all test sizes, and suggests (i) that at most 8 dimensions are necessary to capture the commonalities between network and features in CiteSeer, and (ii) that having more dimensions to capture either shared or non-shared information is not necessarily useful. In other words, 8 overlapping dimensions seem to capture most of what can be captured by AN2VEC on the CiteSeer dataset, and further increase in dimensions (either overlapping or not) would capture redundant information.

Node classification, on the other hand, does not exhibit any consistent behaviour beyond the reduction in performance as the test size increases. Models with less total dimensions seems to perform slightly better at 0 overlap (though this behaviour is reversed on Cora), but neither the ordering of models by total dimensions nor the effect of increasing overlap are consistent across all conditions. This suggests, similarly to Fig. 6a,
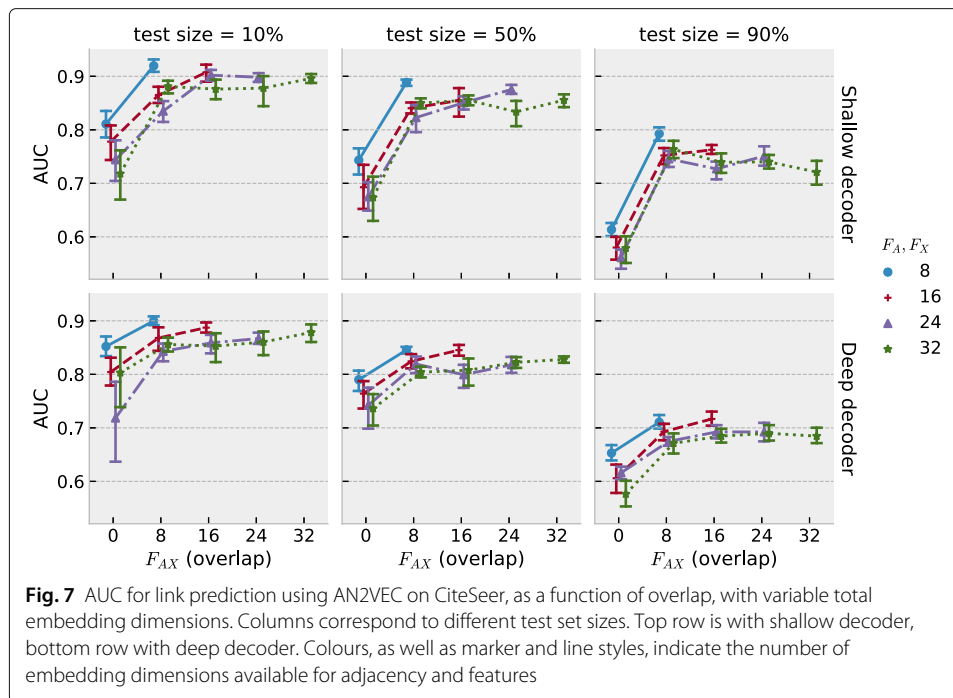
Fig. 6 F1-micro score for node classification using AN2VEC and GraphSAGE over all datasets. AN2VEC top row is the shallow decoder variant, and the bottom row is the deep decoder variant; colour and line styles indicate different levels of overlap. GraphSAGE colours and line styles indicate embedding size as described in the main text (colour and style correspond to the comparable variant of AN2VEC). Each point on a curve aggregates 10 independent training runs. **a** AN2VEC. **b** GraphSAGE

that overlap is less relevant to this particular node classification scheme than it is to link prediction.

Memory usage and time complexity

Finally, we evaluate the resources used by our implementation of the method in terms of training time and memory usage. We use AN2VEC with 100-dimensions intermediate layers in the encoder and the (deep) decoder, with 16 embedding dimensions for each reconstruction task ($F_{\mathbf{A}} + F_{\mathbf{AX}} = F_{\mathbf{X}} + F_{\mathbf{AX}} = 16$), and overlap $F_{\mathbf{AX}} \in \{0, 8, 16\}$. We train that model on synthetic networks generated as in the "Synthetic featured networks" section (setting $\alpha = 0.8$, and without adding any other noise on the features), with $M \in \{50, 100, 200, 500, 1000, 2000, 5000\}$ communities of size $n = 10$ nodes.

Only CPUs were used for the computations, running on a $4 \times$ Intel Xeon CPU E7-8890 v4 server with 1.5 TB of memory. Using 8 parallel threads for training,[12] we record

---

[12]There are actually two levels of threading: the number of threads used in our code for computing losses, and the number of threads used by the BLAS routines for matrix multiplication. We set both to 8, and since both computations alternate this leads to an effective 8 compute threads, with some fluctuations at times.

Fig. 7 AUC for link prediction using AN2VEC on CiteSeer, as a function of overlap, with variable total embedding dimensions. Columns correspond to different test set sizes. Top row is with shallow decoder, bottom row with deep decoder. Colours, as well as marker and line styles, indicate the number of embedding dimensions available for adjacency and features

the peak memory usage,[13] training time, and full job time[14] for each network size, averaged over the three overlap levels. Results are shown in Fig. 9. Note that in a production setting, multiplying the number of threads by *n* will divide compute times by nearly *n*, since the process is aggressively parallelised. A further reduced memory footprint can also be achieved by using sparse encoding for all matrices.

Conclusions

In this work, we proposed an attributed network embedding method based on the combination of Graph Convolutional Networks and Variational Autoencoders. Beyond the novelty of this architecture, it is able to consider jointly network information and node attributes for the embedding of nodes. We further introduced a control parameter able to regulate the amount of information allocated to the reconstruction of the network, the features, or both. In doing so, we showed how shallow versions of the proposed model outperform the corresponding non-interacting reference embeddings on given benchmarks, and demonstrated how this overlap parameter consistently captures joint network-feature information when they are correlated.
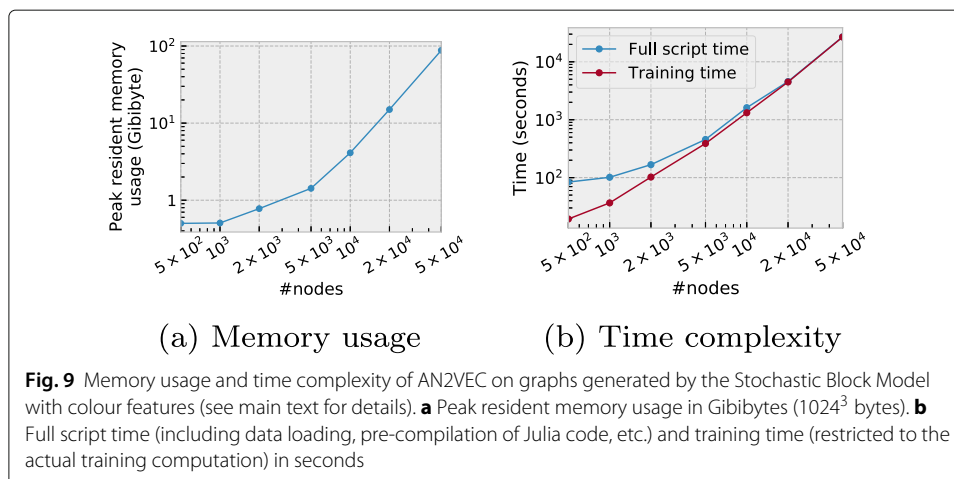
Our method opens several new lines of research and applications in fields where attributed networks are relevant. As an example one can take a social network with the task of predicting future social ties, or reconstructing existing but invisible social ties. Solutions to this problem can rely on network similarities in terms of overlapping sets of common friends, or on feature similarities in terms of common interest, professional or cultural background, and so on. While considering these types of information separately would provide us with a clear performance gain in the prediction, these similarities are not independent. For example, common friends may belong to the same community.

---

[13]Using the `top` utility program.
[14]As reported by our scripts and by GNU Parallel.

Fig. 8 F1-micro score for node classification using AN2VEC on CiteSeer, as a function of overlap, with variable total embedding dimensions. Columns correspond to different test set sizes. Top row is with shallow decoder, bottom row with deep decoder. Colours, as well as marker and line styles, indicate the number of embedding dimensions available for adjacency and features

By exploiting these dependencies our method can provide us with an edge in terms of predictive performance and could indicate which similarities, structural, feature-related, or both, better explain why a social tie exists at all. Another setting where we believe our framework might yield noteworthy insights is when applied to the prediction of side effects of drug pairs (polypharmacy). This problem has recently been approached by Zitnik et al. (2018) by extending GraphSAGE for multirelational link prediction in multimodal networks. In doing so, the authors were able to generate multiple novel candidates of drug pairs susceptible to induce side effects when taken together. Beyond using drug feature vectors to generate polypharmacy edge probabilities, our overlapping encoder units would enable a detailed view on how these side effects occur due to confounding effects of particular drug attributes. It would pinpoint the feature pairs that interacting drugs might share (or not), further informing the drug design process. Furthermore, we

Fig. 9 Memory usage and time complexity of AN2VEC on graphs generated by the Stochastic Block Model with colour features (see main text for details). **a** Peak resident memory usage in Gibibytes (1024^3 bytes). **b** Full script time (including data loading, pre-compilation of Julia code, etc.) and training time (restricted to the actual training computation) in seconds

expect that our method will help yield a deeper understanding between node features and structure, to better predict network evolution and ongoing dynamical phenomena. In particular, it should help to identify nodes with special roles in the network by clarifying whether their importance has structural or feature origin.

In this paper our aim was to ground our method and demonstrate its usefulness on small but controllable featured networks. Its evaluation on more complex synthetic datasets, in particular with richer generation schemes, as well as its application to larger real datasets, are therefore our immediate goals in the future.

Abbreviations
AN2VEC: Attributed node to vector model; AN2VEC-0: Zero overlap model; AN2VEC-16: 16-dimensions overlap model; AN2VEC-S-0: zero overlap model with shallow adjacency decoder; AN2VEC-S-16: 16-dimensions overlap model with shallow adjacency decoder; AP: Average precision; AUC: Area under the ROC curve; Ber: Bernoulli random variable; DANE: Deep attributed network embedding; DW: DeepWalk embedding model; VAE: Variational autoencoder; GAE: Graph autoencoder; GC: Graph convolutional layer; GCN: Graph convolutional network; GCN-VAE: Graph convolutional variational autoencoder; KL: Kullback-Leibler divergence; MLP: Multi-layer perceptron; ReLU: Rectified linuar unit; ROC: Receiver operating characteristic; SC: Spectral clustering; TF/IDF: Term-frequency-inverse-document-frequency; VGAE: Variational graph autoencoder

Authors' contributions
MK, JLA and SL participated equally in designing and developing the project, and in writing the paper. SL implemented the model and experiments. SL and JLA developed and implemented the analysis of the results. All authors read and approved the final manuscript.

Availability of data and materials
The synthetic datasets generated for this work are stochastically created by our implementation, available at github.com/ixxi-dante/an2vec.
The datasets used for standard benchmarking (Cora, CiteSeer, and PubMed) are available at linqs.soe.ucsc.edu/data.
Our implementation of AN2VEC is made using the Julia programming language, and particularly making heavy use of Flux (Innes 2018). Parallel computations were run using GNU Parallel (Tange 2011). Finally, we used StellarGraph (Data61 2018) for the GraphSAGE implementation.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Univ Lyon, ENS de Lyon, Inria, CNRS, UCB Lyon 1, LIP UMR 5668, IXXI, 69007 Lyon, France. [2]Department of Network and Data Science, Central European University, 1051 Budapest, Hungary.

References
Aiken LS, West SG, Reno RR (1994) Multiple Regression: Testing and Interpreting Interactions, vol. 45. Taylor & Francis, New York
Aral S, Muchnik L, Sundararajan A (2009) Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proc Natl Acad Sci 106(51):21544–21549
Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10)
Bojchevski A, Günnemann S (2017) Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. arXiv:1707.03815. Accessed 21 Jan 2019
Bronstein M, Bruna J, LeCun J, Szlam A, Vandergheynst P (2017) Geometric deep learning: Going beyond euclidean data. IEEE Signal Proc Mag 34(4):18–42. https://doi.org/10.1109/MSP.2017.2693418
Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral Networks and Locally Connected Networks on Graphs. arXiv:1312.6203
Cai H, Zheng VW, Chang K (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637. https://doi.org/10.1109/tkde.2018.2807452

Cao S, Lu W, Xu Q (2015) Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. CIKM '15. ACM, New York. pp 891–900. https://doi.org/10.1145/2806416.2806512

Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI'16. AAAI Press, Phoenix. pp 1145–1152

Cataldi M, Caro LD, Schifanella C (2010) Emerging topic detection on twitter based on temporal and social terms evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining. ACM. p 4. https://doi.org/10.1145/1814245.1814249

Chen Z, Badrinarayanan V, Lee C-Y, Rabinovich A (2018) GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In: Dy J, Krause A (eds). Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80. PMLR, Stockholmsmässan. pp 794–803

Data61 C (2018) StellarGraph Machine Learning Library. GitHub. https://github.com/stellargraph/stellargraph

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds). Advances in Neural Information Processing Systems 29. Curran Associates, Inc., Red Hook. pp 3844–3852

Duvenaud DK, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional Networks on Graphs for Learning Molecular Fingerprints. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds). Advances in Neural Information Processing Systems 28. Curran Associates, Inc., Red Hook. pp 2224–2232

Fortunato S (2010) Community detection in graphs. Phys Rep 486(3-5):75–174

Fortunato S, Hric D (2016) Community detection in networks: A user guide. Phys Rep 659:1–44

Gan G, Ma C, Wu J (2007) Data Clustering: Theory, Algorithms, and Applications, vol. 20. Siam, New York

Gao H, Huang H (2018) Deep Attributed Network Embedding. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18). IJCAI-18. International Joint Conferences on Artificial Intelligence, CA, USA. pp 3364–3370

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence And Statistics. PMLR, Sardinia. pp 249–256

Granovetter MS (1977) The Strength of Weak Ties. In: Leinhardt S (ed). Social Networks. Academic Press, Cambridge. pp 347–367. https://doi.org/10.1016/B978-0-12-442450-0.50025-0

Grover A, Leskovec J (2016) Node2vec: Scalable feature learning for networks. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. ACM, New York. pp 855–864. https://doi.org/10.1145/2939672.2939754

Gumperz JJ (2009) The speech community. Linguist Anthropol A Read 1:66

Hamilton W, Ying Z, Leskovec J (2017) Inductive Representation Learning on Large Graphs. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds). Advances in Neural Information Processing Systems 30. Curran Associates, Inc., Red Hook. pp 1024–1034

Holland PW, Laskey K, Leinhardt S (1983) Stochastic blockmodels: First steps. Soc Netw 5(2):109–137

Hours H, Fleury E, Karsai M (2016) Link prediction in the twitter mention network: impacts of local structure and similarity of interest. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference On. IEEE. pp 454–461. https://doi.org/10.1109/icdmw.2016.0071

Innes M (2018) Flux: Elegant machine learning with julia. J Open Source Softw. https://doi.org/10.21105/joss.00602

Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recog Lett 31(8):651–666

Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization. arXiv:1412.6980

Kingma DP, Welling M (2013) Auto-Encoding Variational Bayes. arXiv:1312.6114

Kipf T, Fetaya E, Wang K-C, Welling M, Zemel R (2018) Neural Relational Inference for Interacting Systems. arXiv:1802.04687, Accessed 20 May 2019

Kipf TN, Welling M (2016a) Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907

Kipf TN, Welling M (2016b) Variational Graph Auto-Encoders. arXiv:1611.07308

Kossinets G, Watts DJ (2006) Empirical analysis of an evolving social network. Science 311(5757):88–90

Kossinets G, Watts DJ (2009) Origins of homophily in an evolving social network. Am J Sociol 115(2):405–450

Kumpula JM, Onnela JP, Saramäki J, Kaski K, Kertész J (2007) Emergence of communities in weighted networks. Phys Rev Lett 99(22):228701

La Fond T, Neville J (2010) Randomization tests for distinguishing social influence and homophily effects. In: Proceedings of the 19th International Conference on World Wide Web. ACM. pp 601–610. https://doi.org/10.1145/1772690.1772752

Leo Y, Fleury E, Alvarez-Hamelin JI, Sarraute C, Karsai M (2016) Socioeconomic correlations and stratification in social-communication networks. J R Soc Interface 13(125). https://doi.org/10.1098/rsif.2016.0598

Levy Abitbol J, Karsai M, Fleury E (2018a) Location, occupation, and semantics based socioeconomic status inference on twitter. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). pp 1192–1199. https://doi.org/10.1109/ICDMW.2018.00171

Levy Abitbol J, Karsai M, Magué J-P, Chevrot J-P, Fleury E (2018b) Socioeconomic Dependencies of Linguistic Patterns in Twitter: A Multivariate Analysis. In: Proceedings of the 2018 World Wide Web Conference. WWW '18. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva. pp 1125–1134. https://doi.org/10.1145/3178876.3186011. Accessed 23 Jan 2019

Li C, Ma J, Guo X, Mei Q (2017) Deepcas: An end-to-end predictor of information cascades. In: Proceedings of the 26th International Conference on World Wide Web. WWW '17. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva. pp 577–586. https://doi.org/10.1145/3038912.3052643

Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031

Lü L, Zhou T (2011) Link prediction in complex networks: A survey. Physica A Stat Mech Appl 390(6):1150–1170

Mathieu E, Rainforth T, Siddharth N, Teh YW (2018) Disentangling Disentanglement in Variational Auto-Encoders. arXiv:1812.02833. Accessed 30 Jan 2019

McPherson M, Smith-Lovin L, Cook JM (2001) Birds of a feather: Homophily in social networks. Ann Rev Soc 27(1):415–444

Muzellec B, Cuturi M (2018) Generalizing Point Embeddings using the Wasserstein Space of Elliptical Distributions. arXiv:1805.07594. Accessed 24 Jan 2019

Nair V, Hinton GE (2010) Rectified Linear Units Improve Restricted Boltzmann Machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. ICML'10. Omnipress, Madison. pp 807–814. event-place: Haifa, Israel

Namata G, London B, Getoor L, Huang B (2012) Query-driven active surveying for collective classification. In: 10th International Workshop on Mining and Learning With Graphs, Edinburgh

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830

Peixoto TP (2014) Hierarchical block structures and high-resolution model selection in large networks. Phys Rev X 4(1):011047

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '14. ACM, New York. pp 701–710. https://doi.org/10.1145/2623330.2623732

Rezende DJ, Mohamed S, Wierstra D (2014) Stochastic Backpropagation and Approximate Inference in Deep Generative Models. arXiv:1401.4082

Rosvall M, Axelsson D, Bergstrom CT (2009) The map equation. Eur Phys J Spec Top 178(1):13–23

Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–93. https://doi.org/10.1609/aimag.v29i3.2157

Shen E, Cao Z, Zou C, Wang J (2018) Flexible Attributed Network Embedding. arXiv:1811.10789, Accessed 10 Dec 2018

Shrum W, Cheek Jr NH, MacD S (1988) Friendship in school: Gender and racial homophily. Sociol Educ:227–239. https://doi.org/10.2307/2112441

Tang L, Liu H (2011) Leveraging social media networks for classification. Data Min Knowl Discov 23(3):447–478. https://doi.org/10.1007/s10618-010-0210-x. Accessed 28 Feb 2019

Tange O (2011) Gnu parallel - the command-line power tool. ;login: USENIX Mag 36(1):42–47

Tenenbaum JB, Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323. https://doi.org/10.1126/science.290.5500.2319. http://science.sciencemag.org/content/290/5500/2319.full.pdf

Tran PV (2018) Multi-Task Graph Autoencoders. arXiv:1811.02798. Accessed 9 Jan 2019

Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. ACM, New York. pp 1225–1234. https://doi.org/10.1145/2939672.2939753

Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18. ACM, New York. pp 974–983. https://doi.org/10.1145/3219819.3219890

Zhu D, Cui P, Wang D, Zhu W (2018) Deep variational network embedding in wasserstein space. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18. ACM, New York. pp 2827–2836. https://doi.org/10.1145/3219819.3220052

Zitnik M, Agrawal M, Leskovec J (2018) Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics 34(13):457–466

Publisher's Note