Hyperbolic embedding of attributed and directed networks

McDonald, David; He, Shan

DOI:
10.1109/TKDE.2022.3188426

Hyperbolic Embedding of Attributed and Directed Networks

David McDonald and Shan He

Abstract—Network embedding – finding a low dimensional representation of the nodes with attributes in a hierarchical, directed network remains a challenging problem in the machine learning community. An emerging approach is to embed complex networks – networks of real-world systems – into hyperbolic space due to the fact that hyperbolic space can better naturally represent such a network's hierarchical structure. Existing hyperbolic embedding approaches, however, cannot handle the embedding of attributed directed networks to an arbitrary embedding dimension. To fill this gap, we introduce HEADNet, for Hyperbolic Embedding of Attributed Directed Networks, an algorithm based on extending previous works for embedding directed attributed networks to Gaussian distributions in hyperbolic space of arbitrary dimension. Through experimentation on a variety of both synthetic and real-world networks, we show that HEADNet can achieve competitive performance on common downstream machine learning tasks, including predicting directed links for previously unseen nodes. HEADNet provides an inductive hyperbolic embedding method for directed attributed networks, which opens the door to hyperbolic manifold learning on a wider range of real-world networks. The source code is freely available at https://github.com/DavidMcDonald1993/HEADNET.

Index Terms—complex networks, directed network embedding, attributed network embedding, hyperbolic embedding

1 Introduction

Complex networks are models of diverse real-world systems [1]. The ubiquity of complex networks in such fields as the study of protein-protein interactions, the world wide web and social interaction modelling, has lead to the study of complex networks emerging as a popular research topic. They are described by the set of entities in the system (nodes) and the relations between them (edges). In many complex networks, edges are *directed* and this is essential information that characterises the system as a whole. In addition to the structural information described by edges, often nodes within a complex network are richly annotated with *attributes*. These node attributes provide additional information about the elements within the network and this information is necessary to understand the role that these elements play within the system [2]. However, when dealing with large networks containing billions of nodes, traditional network-based data mining approaches suffer some drawbacks, such as computational complexity, parallelizability and applicability to downstream machine learning tasks [3]. To overcome these drawbacks, *network embedding* is often used to learn low dimensional representations of the nodes in a large network. By learning dense representations of entities in the network, network embedding is able to denoise the network while preserving the intrinsic structural information [3], further improving performance on downstream tasks over using the raw network representation.

A recent trend in network embedding research is to suppose that the hidden metric space underpinning the formation of many complex networks is, in fact, hyperbolic [4]. A hyperbolic metric space has been shown to explain the scale-free degree distribution observed in real-world networks [5]. Moreover, it can explain the *small-world* effect observed in complex networks [6], help with routing of information packets around the network [4] and explains the implicit trade-off between popularity and similarity that controls a node's connections [1].

Due to the aforementioned advantages of assuming an underlying a hyperbolic metric space, hyperbolic space network embedding has received considerable attention. As reviewed in section 1.1, researchers, including us, have proposed several algorithms to embed undirected networks, including attributed networks, to hyperbolic space. However, as yet, there is no approach that can embed attributed directed networks – the most general form of complex networks modelling real world systems – into hyperbolic space. In order to address this, we propose HEADNet to learn hyperbolic representations of arbitrary low dimensions of the nodes in an attributed and directed network. Moreover, we aim to capture the uncertainty in the learned embeddings using Gaussian distribution, similar to [7], but in hyperbolic space. This allows us to characterise node neighbourhood diversity and use an established asymmetric similarity measure with interpretations in both probability and information theory. To this end, we propose both a node embedding model to map attributed nodes to Gaussian distributions in hyperbolic space; and a node similarity measure in hyperbolic space based on a novel mapping procedure to map hyperbolic Gaussian distributions to Euclidean ones, preserving both hyperbolic distance and direction. HEADNet not only learns low-dimensional representations, but also preserves the network's latent structural hierarchy; the interplay of attributes and structural information; and the direction of relations between nodes in a network for downstream inference tasks, allowing for greater insight into complex systems than previously possible.

D. McDonald (ORCID: 0000-0002-0540-8254) works for AIA Insights Ltd. and S. He (ORCID: 0000-0003-1694-1465) is with the Department of Computer Science, University of Birmingham, Birmingham, UK.
Contact e-mail for D. McDonald: davemcdonald93@gmail.com
Manuscript received September 15, 2022; revised xx xx, xxxx.

1.1 Related Work

One popular hyperbolic embedding model is the Popularity-Similarity (or PS) model [1]. This model extends the "popularity is attractive" aphorism of preferential attachment [8] to include node similarity as a second dimension of attachment. HyperMap (HM) [6], a pioneering PS model embedding method, employed maximum likelihood (ML) to search the space of all PS models with similar structural properties as the observed network, to find the one that fit the original network best. Due to the computationally demanding task of ML estimation, heuristic methods were developed to improve upon existing PS model embedding techniques. For example, LABNE [9] uses Laplacian Eigenmaps was to efficiently estimate the angular coordinates of nodes in the PS model. The same authors later extended LABNE with LABNE+HM, which combined both exact and heuristic approaches to leverage the performance of ML estimation against the efficiency of heuristic search, with the trade-off defined with a user controlled parameter [10].

The n-dimensional Poincaré ball provides more degrees of freedom in the embedding process and capture further dimensions of attractiveness than just "popularity" and "similarity". Since trees can be embedded in hyperbolic space without distortion [11] so, some works by embed general graphs to trees and then compute an exact distortion-free embedding of the resulting tree [12]. The algorithm proposed in Nickel and Kiela [13] embeds networks to the Poincaré ball by optimising an objective that maximises the likelihood of observing true node pairs versus arbitrary pairs of nodes in the network. Through this approach, the authors were able to embed hierarchical text corpora such that the semantics of word hierarchy was preserved. This work was later translated to the hyperboloid model [14] and extended to embed any arbitrary parameterised object [15]. Furthermore, hyperbolic skip-gram models have been proposed [16], [17], and heterogeneous networks embedded to hyperbolic space [18]. Interestingly, Wilson et *al.* [19] were able to determine the appropriate curvature to embed any symmetric dissimilarity data, selecting the most appropriate of Spherical, Euclidean or Hyperbolic based on the data. Neural network operations [20] and graph convolution [21] have both recently been generalised to hyperbolic space to allow for non-Euclidean representation learning on attributed networks using both Poincaré and hyperboloid manifolds. Recently, MuRP was proposed to embed multi-relational graphs to the Poincaré ball [22] and Gulcehre et *al.* [23] exploit hyperbolic geometry to reason about arbitrarily deep neural networks. For directed networks, Wu et *al.* [24] construct a bi-partite network from a given directed graph, where each node is transformed into two nodes in the newly constructed network (for incoming edges outgoing edges respectively) and then embed both representations to the PS model. Directed acyclic graphs (DAGs) can be embedded to the Poincaré disk [25], [26]. Of course, transforming general complex networks into DAGs naturally results in a loss of information. Recently, heterogeneous networks have been embedded to hyperbolic space [18]. Previously, we proposed HEAT [27], an undirected hyperbolic network embedding algorithm based on random walks. Furthermore, the Variational Autoencoder (VAE) [28] has recently been derived on the Poincaré disk [29].

Graph2Gauss (G2G) [7] provides an inductive framework to learn Gaussian node representations for attributed networks, based on learning a Kullback-Leibler divergence based objective. This algorithm in particular serves as a great inspiration for our work.

By generalising the convolution operation from regular pixel lattices to arbitrary graphs, it is possible embed and classify entire small graphs [30], [31]. The popular Graph Convolutional Network (GCN) [32] extend and simplify existing graph convolution approaches to embed nodes in a semi-supervised setting. Furthermore, GraphSAGE [2] introduces an inductive framework for online learning of node embeddings capable of generalising to unseen nodes. Geom-GCN [33] overcomes the drawback of traditional message-passing neural networks (like GCNs) that the structural information of node neighbourhoods is lost in graph-invariant aggregation through the use of a novel aggregation function based on node embedding to exploit the continuous space underlying the graph. Text-assisted Deepwalk (TADW) [34] generalises Deepwalk [35] to nodes with text attributes. RoSANE [36] introduces a scalable framework based on the Skip-Gram model for embedding sparse attributed networks. Also, GloDyNE [37] adapts Skip-Gram for network embedding in a dynamic setting. LANE [38] can deal with weighted/unweighted networks and binary/real valued attributes and labels. Other approaches are based on distributions of single and pairs of attributes [39]; some based on known community structure [40]; while others draw from the well known fields of manifold learning and multi-view learning to align the projections based on topology and attributes [41].

Many random walk based approaches can handle directed networks and are capable of an asymmetric distance measure when both the "source" and "context" embeddings are retained [35], [42]. For unattributed directed network embedding LINE (with second-order proximity) is able to handle directed graphs by assuming that "vertices sharing many connections to other vertices are similar to each other" [43]. Furthermore, Asymmetric Transitivity Preservation (ATP) [44] aims to directly preserve the inherently asymmetric nature of the transitive relationships in Community Question Answering graphs, by learning two representations of each node in the network.

2 Hyperboloid Model

Hyperbolic space cannot be embedded into Euclidean space without distortion [45]. Due to the fundamental difficulty of representing spaces of constant negative curvature as subsets of Euclidean spaces, there are not one but many equivalent models of hyperbolic spaces. The models are equivalent because they can be freely mapped to each other by an distance preserving map – called an *isometry*. Each model emphasises different aspects of hyperbolic geometry, but no model simultaneously represents them all.

We select the *hyperboloid* model due its rising popularity in the literature, which is largely down to the comparatively simple form of all Riemannian operations [14], [46] compared with the more traditional ball models. Both the *Poincaré* and *Klein* models of hyperbolic space are formed as projections of the points from the hyperboloid to balls orthogonal to the main axis of the hyperboloid [47]. Informally, we can see these relationships as analogous to the relationship between a projected map and a globe [45]. For our purposes, the hyperboloid model allows for a more simple formulation of Riemannian operators, such as the exponential map, logarithmic map, and parallel transport that will be leveraged later in this paper.

Unlike disk models that sit in an ambient Euclidean space of dimension n, the hyperboloid model of n-dimensional hyperbolic geometry sits in $n+1$-dimensional Minkowski space-time. Furthermore, the set of hyperboloid points do not form a disk in this ambient space, but an n-dimensional hyperboloid. $n+1$-dimensional Minkowski space-time (denoted $\mathbb{R}^{n:1}$) is defined as the concatenation of n-dimensional Euclidean space with an additional time co-ordinate t. A vector $\mathbf{u} \in \mathbb{R}^{n:1}$ has spacial coordinates $\mathbf{u}^i$ for $i = 1, 2, ..., n$ and time coordinate $\mathbf{u}^{n+1}$. For two vectors in Minkowski space $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{n:1}$, the Minkowski bilinear form $\langle\cdot\rangle_{\mathbb{R}^{n:1}}$ is defined as:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}} := \sum_{i=1}^{n} \mathbf{u}^i \mathbf{v}^i - \psi^2 \mathbf{u}^{n+1} \mathbf{v}^{n+1} \tag{1}$$

where ψ is the speed of information flow in the system (often set to 1 for simplified calculations) [48]. This bilinear form serves as an inner product and allows one to compute vector norms in a familiar way:

$$||\mathbf{u}||_{\mathbb{R}^{n:1}} := \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}_{\mathbb{R}^{n:1}} \tag{2}$$

It is worth noting that $n+1$-dimensional Minkowski space-time contains the entire n-dimensional Euclidean space: $\mathbb{R}^n \subset \mathbb{R}^{n:1}$ since $\mathbb{R}^n$ is the set of all $\mathbf{x} \in \mathbb{R}^{n:1}$ such that its time co-ordinate $\mathbf{x}^{n+1} = 0$. Further, the Minkowski bilinear form (eq. (1)) is a generalisation of the Euclidean inner product and is equivalent for all $\mathbf{x} \in \mathbb{R}^n = \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \mathbf{x}^{n+1} = 0\}$. The points $\mathbf{u} \in \mathbb{R}^{n:1}$ satisfying: $\mathbb{H}^n := \{\mathbf{u} \in \mathbb{R}^{n:1} \mid \langle \mathbf{u}, \mathbf{u} \rangle_{\mathbb{R}^{n:1}} = -1 \wedge \mathbf{u}^{n+1} > 0\}$ define the hyperboloid model [46]. The first condition defines a hyperbola of two sheets, and the second condition selects the top sheet. Shortest paths (*geodesics*) between points on the model are given by the hyperbola formed by the intersection of $\mathbb{H}^n$ and the two dimensional plane containing the origin of Minkowski space ($\mathbf{0} \in \mathbb{R}^{n:1}$) and both of the points [45]. The distance along the geodesic between two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$ is given by $d_{\mathbb{H}^n}(\mathbf{u}, \mathbf{v}) = \operatorname{arccosh}(-\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbb{R}^{n:1}})$ and is analogous to the length of segment of a great circle that connects two points in spherical geometry [45]. The tangent space $T_{\mathbf{u}}\mathbb{H}^n$ of a vector $\mathbf{u} \in \mathbb{H}^n$ is defined as $T_{\mathbf{u}}\mathbb{H}^n := \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \langle \mathbf{u}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}} = 0\}$ and is the collection of all points in $\mathbb{R}^{n:1}$ that are orthogonal to $\mathbf{u}$ with respect to the Minkowski bilinear form. It can be shown that $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{R}^{n:1}} > 0$, for all $\mathbf{x} \in T_{\mathbf{u}}\mathbb{H}^n$, for all $\mathbf{u} \in \mathbb{H}^n$ [45]. In other words, the tangent space of the hyperboloid is positive definite (with respect to the Minkowski bilinear form) for all points on the hyperboloid and so $\mathbb{H}^n$ (equipped with eq. (1)) satisfies the requirements of a Riemannian manifold [45].

3 HEADNet Algorithm

HEADNet is built upon the principle that nodes in an attributed directed network can be mapped to Gaussian distributions in hyperbolic space. As discussed previously, hyperbolic space is selected since it can better reflect the inherent hierarchy of the elements within many real-world systems. The overall intuition behind this approach is that the distributions capture the uncertainty in the learned representations of nodes. Uncertainty here means that nodes with highly diverse neighbourhoods – in terms of attribute presence and/or class label, for example – tend to be embedded to distributions with greater variance [7]. Kullback-Leibler (KL) divergence, a measure of the penalty of encoding one distribution as another, is used as an asymmetric measure of similarity between nodes and forms the basis of an objective function that aims to maximise the similarity between the directed edges in the network, while minimising the similarity between all other node pairs.

3.1 Problem Definition

We consider a *directed* network of N nodes given by the set V with $|V| = N$. $E := \{(u, v)\} \subseteq V \times V$ denotes the set of all interactions between the nodes. Since the network is directed, $(u, v) \in E \not\Rightarrow (v, u) \in E$. Further, the nodes in V may be annotated with d-dimensional attributes described in matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$. In the special case that nodes do not have attributes, we set $\mathbf{X} = \mathbf{I}_N$ which is the N-dimensional identity matrix. We consider the problem of representing the network given as $\mathbb{G} = (V, E, \mathbf{X})$ as a set of n-dimensional Gaussian distributions in hyperbolic space, such that asymmetric measure of similarity between nodes in the network is preserved, where $n \ll N$. The distributions are described by $n+1$-dimensional hyperboloid mean vectors, along with n-dimensional Euclidean vectors corresponding to diagonal co-variance matrices.

3.2 HEADNet Algorithm Overview

HEADNet is comprised of the following three main steps:

3.2.1 *Step 1: Network Embedding*

This step addresses the problem described in section 3.1 by introducing a novel node embedder to transform the nodes in $\mathbb{G}$ into a set of N n-dimensional Gaussian distributions in hyperboloid model of hyperbolic space. Edges in E are not required to perform an embedding, however they are used to train the embedder.

3.2.2 *Step 2: Node Similarity Measurement*

For an arbitrary pair of nodes in a complex network, this step transports their hyperbolic distributions provided by step 1 as input and provides a asymmetric measure of similarity between them that is assumed to be proportional to their connection probability.

3.2.3 *Step 3: Node Representation Learning*

This step uses the output of step 2 as part of a learning objective explicitly based on known connectivity information to train the embedder model used in step 1. The learning objective is designed to maximise the similarity between all of the node pairs $(u, v) \in E$ and minimise the similarity of all other pairs of nodes in $V \times V \setminus E$.

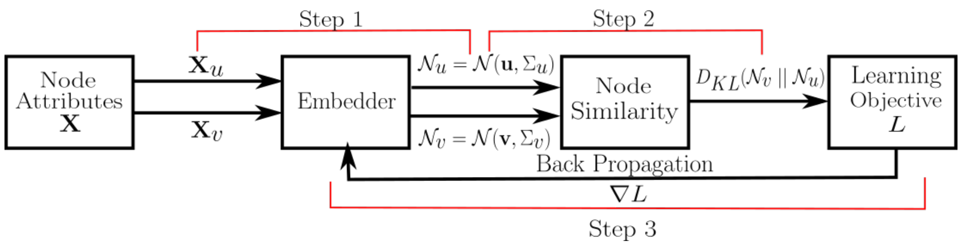

Step 1
Step 2
Node Attributes $\mathbf{X}$
$\mathbf{X}_u$
$\mathbf{X}_v$
Embedder
$\mathcal{N}_u = \mathcal{N}(\mathbf{u}, \Sigma_u)$
$\mathcal{N}_v = \mathcal{N}(\mathbf{v}, \Sigma_v)$
Node Similarity
$D_{KL}(\mathcal{N}_v \,||\, \mathcal{N}_u)$
Learning Objective L
Back Propagation
∇L
Step 3

Fig. 1: Schematic representation of the main steps of HEADNet. The first step is a node embedder that accepts a d-dimensional vector, $\mathbf{X}_u \in \mathbb{R}^d$, corresponding to attributes of a node u in the network, and outputs a Gaussian distribution for node u in hyperbolic space, denoted $\mathcal{N}_u$. The distributions are parameterised by n-dimensional hyperbolic means $\mathbf{u} \in \mathbb{H}^n$ and n-dimensional Euclidean vectors Σ_u, corresponding to diagonal variance matrices. Next, the node similarity step takes two hyperbolic Gaussian distributions $\mathcal{N}_u$ and $\mathcal{N}_v$, output by the node embedder, and computes a asymmetric node similarity measure $D_{KL}(\mathcal{N}_v \,||\, \mathcal{N}_u)$ using Kullback-Leibler divergence. Finally, node similarity is used in as part of a loss function L that aims to maximise the similarity of the directed edges comprising the edge set of the network $(u, v) \in E$. The objective function is used to drive the learning of embedder parameters through back-propagation based on ΔL. Full details are provided in section 3.3, section 3.4 and section 3.5.

3.2.4 *Combining the Steps*

Figure 1 provides a high-level overview of these three steps. Once the embedder has been trained using step 3, it can then be used by itself to map nodes to distributions in hyperbolic space (using step 1 only), or it can be used in conjunction with the node similarity step to query the similarly of any arbitrary pair of nodes (combining steps 1 and 2 to map those nodes to low-dimensional distributions and then measure the similarity using KL divergence). For both applications, the nodes do not necessarily have to be previously seen during model training, although, in this case, the nodes are assumed to be annotated with attributes.

3.3 Network Embedding Step

This subsection describes how node attributes $\mathbf{X}_u$ are mapped to a pair of vectors $(\mathbf{u}, \Sigma_u) \in \mathbb{H}^n \times \mathbb{R}^n$ describing a Gaussian distribution in hyperbolic space where $\mathbf{u}$ is an $n+1$-dimensional vector on the hyperboloid model representing the mean of the distribution describing node u, and Σ_u is an n-dimensional Euclidean vector that represents the diagonal co-variance matrix of the distribution for node u. We compute $\mathbf{u}$ and Σ_u as functions of node attributes $\mathbf{X}_u$. Mapping directly from attributes has the advantage of readily handling previously unseen nodes [7].

To this end we propose the following node embedder model: Following previous works [7], our embedder is a two-layer feed-forward neural network with two parallel output layers. This architecture provides a flexible non-linear mapping between attributes and distributions, while reducing the overall number of parameters compared to two separate models for computing means and variances respectively. The embedder has three components: a map from attributes $\mathbf{X}_u$ to $\mathbf{h}_u$, a $d' \ll d$ dimensional hidden layer representation; map from the hidden representation $\mathbf{h}_u$ to Euclidean vectors representing diagonal co-variance matrices $\Sigma_u \in \mathbb{R}^n$; and a map from the hidden representation $\mathbf{h}_u$ to hyperbolic means $\mathbf{u} \in \mathbb{H}^n$.

3.3.1 *Mapping to Hidden Representation $\mathbf{h}_u$*

We first map non-linearly from node attributes to a latent hidden representation space $\mathbf{h}_u \in \mathbb{R}^{d'}$ ($d' \ll N$) using $\mathbf{h}_u = \mathrm{relu}\left(\mathbf{W}_h \mathbf{X}_u + \mathbf{b}_h\right)$ where $\mathbf{W}_h \in \mathbb{R}^{d' \times d}$ and $\mathbf{b}_h \in \mathbb{R}^{d'}$ are a weight matrix and bias to learn. We share this intermediary representation $\mathbf{h}_u$ for computing both $\mathbf{u}$ and Σ_u, which has the advantage of regularising the model by reducing the overall number of parameters [7]. We select the relu function as a non-linearity to overcome the vanishing gradients problem and due to its superior performance on some small preliminary benchmark tasks.

3.3.2 *Mapping from $\mathbf{h}_u$ to Co-variance Vector Σ_u*

From the hidden representation of node u, $\mathbf{h}_u$, we compute its co-variance vector Σ_u using $\Sigma_u = \mathrm{elu}_{\alpha=1}\left(\mathbf{W}_\Sigma \mathbf{h}_u + \mathbf{b}_\Sigma\right) + 1$. Recall that the elu function is bounded in the range $[-1, \infty]$, and so we add one to bound the output in the range $[0, \infty]$ and thus ensure positive definiteness in Σ as required [7].

3.3.3 *Mapping from $\mathbf{h}_u$ to Hyperbolic Mean Vector $\mathbf{u}$*

To map from the Euclidean hidden representation $\mathbf{h}_u \in \mathbb{R}^{d'}$ to a hyperbolic mean vector $\mathbf{u} \in \mathbb{H}^n$, we apply a fundamental operation of Riemannian geometry, the exponential map, and leverage a property of the origin of the hyperboloid.

To transport a vector $\mathbf{x} \in T_\mathbf{u}\mathbb{H}^n$ from the tangent space of a point $\mathbf{u} \in \mathbb{H}^n$ on the hyperboloid to a vector $\mathbf{v} \in \mathbb{H}^n$ on the hyperboloid itself, one uses the *exponential map* $\mathrm{Exp}_\mathbf{u}(\mathbf{x}) : T_\mathbf{u}\mathbb{H}^n \to \mathbb{H}^n$ [46], defined as:

$$\mathbf{v} = \mathrm{Exp}_\mathbf{u}(\mathbf{x}) = \cosh(r)\mathbf{u} + \sinh(r)\mathbf{x}/r \tag{3}$$

where $r = ||\mathbf{x}||_{\mathbb{R}^{n:1}}$ is the Minkowski norm of $\mathbf{x}$ (see eq. (2)).

The *origin* of the hyperboloid (informally, its "bottom tip") is defined as: $\mu_\mathbf{0} := [0, ..., 0, 1]^T \in \mathbb{H}^n$ and is the point on the upper hyperboloid with all spacial coordinates set to 0 and a time coordinate of 1. The following property of the tangent space of this point follows from definition:

$$T_{\mu_\mathbf{0}}\mathbb{H}^n = \{\mathbf{x} \in \mathbb{R}^{n:1} \mid \langle \mathbf{x}, \mu_\mathbf{0} \rangle_{\mathbb{R}^{n:1}} = 0\} \equiv \mathbb{R}^n. \tag{4}$$

That is to say that the tangent space of the origin of the hyperboloid, $T_{\mu_\mathbf{0}}\mathbb{H}^n$, is the entire n-dimensional Euclidean space (every point in $n+1$-dimensional Minkowski space with a 0 time co-ordinate). Furthermore, in this space, Euclidean and Minkowski norms coincide: $\forall \mathbf{x} \in T_{\mu_\mathbf{0}}\mathbb{H}^n =$

$\mathbb{R}^n$, $||\mathbf{x}||_{\mathbb{R}} \equiv ||\mathbf{x}||_{\mathbb{R}^{n:1}}$. We combine both $\text{Exp}_{\mathbf{u}}(\cdot)$ (eq. (3)) and this property of the origin of the hyperboloid ($T_{\mu_0}\mathbb{H}^n = \mathbb{R}^n$) to define a point-wise non-linearity on the output of a dense neural network layer: $\text{Exp}_{\mu_0} : \mathbb{R}^n \to \mathbb{H}^n$, computed as:

$$\text{Exp}_{\mu_0}(\mathbf{x}) = [\sinh(||\mathbf{x}||)\,\mathbf{x}/||\mathbf{x}||, \cosh(||\mathbf{x}||)] \qquad (5)$$

for $\mathbf{x} \in \mathbb{R}^n$, where $||\cdot||$ is the Euclidean norm operation.

Returning to the node embedder model, to compute the hyperbolic mean vector $\mathbf{u} \in \mathbb{H}^n$ from the hidden representation $\mathbf{h}_u$, we propose the use of $\mathbf{u} = \text{Exp}_{\mu_0}(\mathbf{W}_{\mathbb{H}^n}\mathbf{h}_u + \mathbf{b}_{\mathbb{H}^n})$ where $\mathbf{W}_{\mathbb{H}^n} \in \mathbb{R}^{n\times d'}$ and $\mathbf{b}_{\mathbb{H}^n} \in \mathbb{R}^n$ are a weight matrix and bias to learn. The use of Exp_{μ_0} as a non-linearity provides a differentiable mapping from n-dimensional Euclidean space ($\mathbb{R}^n = T_{\mu_0}\mathbb{H}^n$) to $\mathbb{H}^n$ as required.

3.4 Node Similarity Measurement

This subsection describes the procedure for computing the similarity measure for an arbitrary pair of nodes, using the Gaussian distributions provided by the embedder. It is largely inspired by the wrapped Gaussian distribution in hyperbolic space introduced in [49]. We use Kullback-Leibler divergence as the measure of similarity between node distributions. For continuous distributions P and Q, Kullback-Leibler divergence $D_{KL}(P\,||\,Q)$ is given by:

$$D_{KL}(P\,||\,Q) := \int_x P(x)\log\left(P(x)/Q(x)\right) \qquad (6)$$

We see immediately that eq. (6) is asymmetric in P and Q. Furthermore, when P and Q are normal distributions, it has been shown that eq. (6) simplifies to:

$$D_{KL}(P\,||\,Q) = 0.5 * \left[tr\left(\Sigma_u^{-1}\Sigma_v\right) + \right.$$
$$\left.(\mu_u - \mu_v)^T\,\Sigma_u^{-1}(\mu_u - \mu_v) - n - \log\left(det\left(\Sigma_v\right)/det\left(\Sigma_u\right)\right)\right] \qquad (7)$$

where $tr(.)$ and $det(.)$ denote the trace and determinant of of a matrix respectively [7].

3.4.1 Mapping Hyperbolic To Euclidean Coordinates

Since the node embedder outputs Euclidean co-variance matrices, we only require a slight modification to the term $(\mu_u - \mu_v)^T\,\Sigma_u^{-1}(\mu_u - \mu_v)$ in order to apply eq. (7). To this end, we propose a mapping from the hyperbolic mean vectors output by the embedder to Euclidean vectors, such that hyperbolic relationships are preserved. This mapping preserves the hyperbolic distance between a given node pair, while allowing for a simple form of the training objective (see eq. (12)). For the remainder of this section, we will introduce a mapping $\Psi_{\mathbf{u}}$ that will replace the term $(\mu_u - \mu_v)^T\,\Sigma_u^{-1}(\mu_u - \mu_v)$ to reflect a hyperbolic distribution, rather than a Euclidean one. $\Psi_{\mathbf{u}}$ will behave exactly like the original term in that it will be a weighted sum over a distance vector. Furthermore, when all variances are identity matrices, then our derived equivalent to eq. (7) will become equivalent to hyperbolic distance squared as expected. This makes our approach suitable for undirected networks.

For a node pair (u, v), the mapping, Ψ, is a two step procedure that relies upon $\mathbf{u}$ – the hyperbolic mean of node u – to serve as an "anchor point". The two steps are: first, transport vectors from $\mathbb{H}^n$ to the tangent space of the anchor point $T_{\mathbf{u}}\mathbb{H}^n$; then, transport vectors from $T_{\mathbf{u}}\mathbb{H}^n$ to the tangent space of the origin of the hyperboloid $T_{\mu_0}\mathbb{H}^n$: n-dimensional Euclidean space (see eq. (4)).

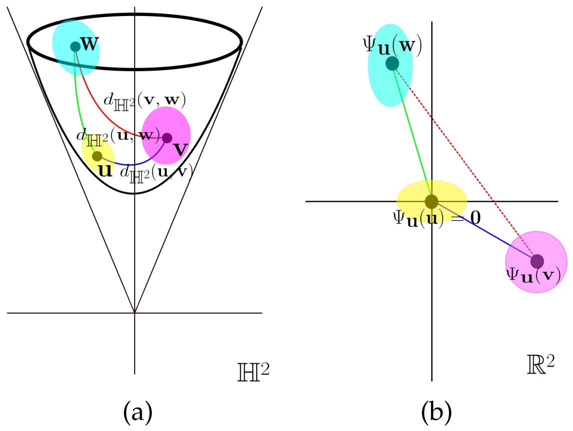

w
$d_{\mathbb{H}^2}(\mathbf{v},\mathbf{w})$
$d_{\mathbb{H}^2}(\mathbf{u},\mathbf{w})$
v
u
$d_{\mathbb{H}^2}(\mathbf{u},\mathbf{v})$
$\mathbb{H}^2$
(a)
$\Psi_{\mathbf{u}}(\mathbf{w})$
$\Psi_{\mathbf{u}}(\mathbf{u}) = 0$
$\Psi_{\mathbf{u}}(\mathbf{v})$
$\mathbb{R}^2$
(b)

Fig. 2: Demonstration of relative distance-preserving property of Ψ operator. (a) shows three points $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{H}^2$ on the two-dimensional hyperboloid. The three lines connecting the points show the geodesics between the points with lengths of $d_{\mathbb{H}^2}(\mathbf{v},\mathbf{w})$, $d_{\mathbb{H}^2}(\mathbf{u},\mathbf{w})$, and $d_{\mathbb{H}^2}(\mathbf{u},\mathbf{v})$. (b) plots their resulting positions in $\mathbb{R}^2$ after applying $\Psi_{\mathbf{u}}$. Distances $d_{\mathbb{H}^2}(\mathbf{u},\mathbf{v})$ and $d_{\mathbb{H}^2}(\mathbf{u},\mathbf{w})$ are preserved but $d_{\mathbb{H}^2}(\mathbf{v},\mathbf{w})$ is not.

For two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^n$, we can map to Euclidean space ($T_{\mu_0}\mathbb{H}^n$) by combining the logarithmic map operation of the hyperboloid, defined as:

$$\mathbf{x} = \text{Log}_{\mathbf{u}}(\mathbf{v}) = \text{arccosh}(\alpha)(\mathbf{v} - \alpha\mathbf{u})/\sqrt{\alpha^2 - 1} \qquad (8)$$

and the parallel transport to μ_0 operation, defined as:

$$\mathbf{y} = \underset{\mathbf{u}\to\mu_0}{\text{PT}}(\mathbf{x}) = \mathbf{x} + \langle\mu_0 - \beta\mathbf{u}, \mathbf{x}\rangle_{\mathbb{R}^{n:1}}(\mathbf{u} + \mu_0)/(\beta + 1) \qquad (9)$$

where $\alpha = -\langle\mathbf{u},\mathbf{v}\rangle_{\mathbb{R}^{n:1}}$ and $\beta = -\langle\mathbf{u},\mu_0\rangle_{\mathbb{R}^{n:1}}$. We now introduce $\Psi_{\mathbf{u}}$ as the composition of eq. (8) and eq. (9):

$$\Psi_{\mathbf{u}} : \mathbb{H}^n \to \mathbb{R}^n := \underset{\mathbf{u}\to\mu_0}{\text{PT}} \circ \text{Log}_{\mathbf{u}}. \qquad (10)$$

We apply $\Psi_{\mathbf{u}}$ to map hyperbolic vectors to Euclidean vectors. For a given node pair (u, v), each mapping from their hyperbolic co-ordinates $\mathbf{u}$ and $\mathbf{v}$ to their Euclidean positions $\Psi_{\mathbf{u}}(\mathbf{u})$ and $\Psi_{\mathbf{u}}(\mathbf{v})$ is performed "relative" to the anchor node u. In general, for $\mathbf{x}, \mathbf{y}, \mathbf{v} \in \mathbb{H}^n$, $\mathbf{x} \neq \mathbf{y} \implies \Psi_{\mathbf{x}}(\mathbf{v}) \neq \Psi_{\mathbf{y}}(\mathbf{v})$ and so the Euclidean position of node v (relative to anchor node u), $\Psi_{\mathbf{u}}(\mathbf{v})$, is wholly dependent on hyperbolic coordinate $\mathbf{u}$ of the anchor node u that is used in the mapping. In other words, changing $\mathbf{u}$ would result in a different position for $\Psi_{\mathbf{u}}(\mathbf{v})$. The Euclidean norm of $\Psi_{\mathbf{u}}(\mathbf{v})$ is equal to the hyperbolic distance between the mean vectors of nodes u and v on the hyperboloid [49]. That is $||\Psi_{\mathbf{u}}(\mathbf{v})|| = d_{\mathbb{H}^n}(\mathbf{u},\mathbf{v}) = \text{arccosh}(-\langle\mathbf{u},\mathbf{v}\rangle_{\mathbb{R}^{n:1}})$. This follows from eq. (2), and so we can interpret $\Psi_{\mathbf{u}}$ as a distance preserving map for all nodes relative to node u. For all $\mathbf{u},\mathbf{v},\mathbf{w} \in \mathbb{H}^n$, $\Psi_{\mathbf{u}}$ preserves distances $d_{\mathbb{H}^n}(\mathbf{u},\mathbf{v})$ and $d_{\mathbb{H}^n}(\mathbf{u},\mathbf{w})$, but not $d_{\mathbb{H}^n}(\mathbf{v},\mathbf{w})$ [49]. Further, we observe that for all $\mathbf{u} \in \mathbb{H}^n$, $\Psi_{\mathbf{u}}(\mathbf{u}) = \mathbf{0}$. Figure 2 provides an example Ψ mapping from $\mathbb{H}^2$ to $\mathbb{R}^2$, that shows how $\Psi_{\mathbf{u}}$ preserves hyperbolic distances to node u but distorts all distances between other nodes.

3.4.2 *Node Similarity Measurement*

Before putting everything together, note that the terms involving Σ in eq. (7) simplify to a simple sum over vector elements since Σ is a diagonal matrix [7]. We can now propose the following modified form for eq. (7) as an asymmetric measure of node similarity in hyperbolic space:

$$\underset{KL}{D}(\mathcal{N}_v \,||\, \mathcal{N}_u) = \frac{1}{2} \sum_{i=1}^{n} \left[\Gamma_{uv}^i + \Omega_{uv}^i - 1 - \log\left(\Gamma_{uv}^i\right) \right] \quad (11)$$

where $\mathbf{x}^i$ is the i-th component of vector $\mathbf{x}$, $\Gamma_{uv} := \Sigma_v / \Sigma_u$, and $\Omega_{uv} := \left[\Psi_{\mathbf{u}}(\mathbf{u}) - \Psi_{\mathbf{u}}(\mathbf{v})\right]^2 / \Sigma_u = \Psi_{\mathbf{u}}(\mathbf{v})^2 / \Sigma_u$ since $\Psi_{\mathbf{u}}(\mathbf{u}) = \mathbf{0} \; \forall \mathbf{u} \in \mathbb{H}^n$.

3.5 Node Representation Learning

This subsection describes the learning objective of HEADNet, based on the node similarity computed in eq. (11). The learning procedure is based on the idea that we aim to maximise the similarity between observed node pairs – the directed edges in the network – while simultaneously minimising the similarity between all other node pairs.

3.5.1 *Learning Objective*

Following previous works [7], we adopt an energy-based objective and define the energy $\mathcal{E}_{uv}$ between two nodes u and v (which are described by normal distributions $\mathcal{N}_{\mathbf{u}}$ and $\mathcal{N}_{\mathbf{v}}$ to be simply: $\mathcal{E}_{uv} = D_{KL}(\mathcal{N}_v \,||\, \mathcal{N}_u)$. Note that, in general, $\mathcal{E}_{uv} \neq \mathcal{E}_{vu}$ as required. As with other energy-based methods, we aim to minimise the energy (reduce the KL divergence) for desirable node pairings (the edges in E) and, likewise, maximise the energy of undesirable pairings (the non-edge set $V \times V \setminus E$). Our objective is based on on negative log-likelihood:

$$L = \underset{(u,v)\sim E, S\sim \mathbb{S}_{uv}^k}{\mathbb{E}} \left[\mathcal{E}_{uv} + \log \sum_{(u',v')\in S} \exp\left(-\mathcal{E}_{u'v'}\right) \right] \quad (12)$$

where $\mathbb{S}_{uv}^k := \{S \subseteq V \times V : |S| = k \wedge (u,v) \in S\}$ is the set of all subsets of node pairs containing exactly k elements including the pair (u,v). The first term ($\mathcal{E}_{uv}$) will cause the average energy of all pairs $(u,v) \in E$ to be reduced. The second, contrastive, term $\log \sum_{(u',v')\in S} \exp\left(-\mathcal{E}_{u'v'}\right)$ causes the energy of all undesirable pairs to be "pulled-up" a little [50]. The parameter k is a hyper-parameter used to control the number of undesirable pairs seen for every desirable pair, and is analogous to negative sample number in skip-gram models [13].

4 Experimental Validation

4.1 Network Datasets

4.1.1 *Synthetic Networks*

We generate 30 synthetic scale free directed networks with $N = 1000$ nodes. We set the probability for adding a new node connected to an existing node chosen randomly according to the in-degree distribution to $\alpha = 0.41$. The probability for adding an edge between two existing nodes is $\beta = 0.54$; and the probability for adding a new node connected to an existing node chosen randomly according to the out-degree distribution $\gamma = 0.05$ [51].

TABLE 1: Description of selected benchmark algorithms.

Algorithm	Variant	Description	Metric Space	Directed	Attributes
ATP [44]	ATP (log)	ATP (log transform)	$\mathbb{R}$	Yes	No
LINE [43]	LINE	LINE (2nd order proximity)	$\mathbb{R}$	Yes	No
G2G [7]	G2G$_{\text{NA},K=1}$	G2G, no attributes, $K=1$	$\mathbb{R}$	Yes	No
	G2G$_{\text{NA},K=3}$	G2G, no attributes, $K=3$	$\mathbb{R}$	Yes	No
	G2G$_{K=1}$	G2G, attributes, $K=1$	$\mathbb{R}$	Yes	Yes
	G2G$_{K=3}$	G2G, attributes, $K=3$	$\mathbb{R}$	Yes	Yes
NK [13]	NK	NK Algorithm	$\mathbb{H}$	No	No
Deepwalk [35]	Deepwalk	Random-walk sampling	$\mathbb{R}$	No	No
Sage-GCN [2], [32]	Sage-GCN	Graph Convolutional Network	$\mathbb{R}$	Yes	Yes
HNN [20]	HNN	Hyperbolic Neural Network	$\mathbb{H}$	Yes	Yes
HGCN [21]	HGCN	HyperbolicGCN	$\mathbb{H}$	Yes	Yes
HEADNet	HEADNet$_{\text{NA},\Sigma=\mathbb{I}}$	HEADNet, no attributes, $\Sigma=\mathbb{I}$	$\mathbb{H}$	No	No
	HEADNet$_{\text{NA}}$	HEADNet, no attributes	$\mathbb{H}$	Yes	No
	HEADNet$_{\Sigma=\mathbb{I}}$	HEADNet, attributes, $\Sigma=\mathbb{I}$	$\mathbb{H}$	No	Yes
	HEADNet	HEADNet	$\mathbb{H}$	Yes	Yes

4.1.2 *Real World Networks*

We evaluate HEADNet on four directed citation networks: Cora ML ($N = 2995$, $|E| = 8416$, $d = 2879$, $y = 7$), Citeseer ($N = 4230$, $|E| = 5358$, $d = 2701$, $y = 6$), Pubmed ($N = 18230$, $|E| = 79612$, $d = 500$, $y = 3$), and Cora ($N = 19793$, $|E| = 65311$, $d = 8710$, $y = 70$) [7]; and one unattributed directed social network: Wiki vote ($N = 7115$, $|E| = 103689$, $d = -$, $y = -$) [52], where N, $|E|$, d, and y to denote the number of nodes, edges, attributes and node classes respectively. $d = -$ and $y = -$ denote that a network does not have node attributes and node labels respectively.

4.2 Benchmark Algorithms

Table 1 provides details of all of the benchmark algorithms used in this study. We have selected popular graph embedding algorithms from the literature that that embed nodes to a range of different metric spaces: Euclidean (ATP [44], LINE [43], Deepwalk [35], Sage-GCN [32]), Hyperbolic (NK [13], HGCN [20], HNN [21]), and distribution (G2G [7]) spaces. For all benchmark algorithms, we adopt the default hyper-parameter settings except where stated. For LINE we set the order to second only as the authors claim that this is suitable for directed graphs [43]. We set the negative ratio to 10 and the number of epochs to 1000. For ATP we break cycles using the hierarchical grouping method that the authors demonstrated obtained good results [53]. We select the log transform as it was the best performing [44]. For G2G$_{*,K=\{1,3\}}$, we set K to 1 and 3 respectively and G2G$_{\text{NA},*}$ denotes G2G without attributes. For all G2G models, we fix the hidden dimension to 128 and train for the default maximum of 2000 epochs. For NK we set the negative sample size to 10, the hyperbolic learning rate to 1 and train for 1500 epochs. For Deepwalk we set walks per node to 10, walk length to 80, context size to 10 and top-k value to 30. For Sage-GCN, we use the GraphSAGE implementation [2] of GCN and set learning rate to 0.001, dropout rate to 0.5, batch size to 512, normalisation to true, weight decay rate $1e-4$ and epochs to 100. For HNN and HCGN, we use the PyTorch implementation (available at https://github.com/HazyResearch/hgcn) [21] and set the manifold space to Poincaré, the number of hidden layers to 2, use dropout value of 0.5, and use relu activation functions. For HEADNet we train for 1000 epochs, stopping early if there is no improvement in loss (eq. (12)) after 25 epochs, and set the ratio of negative to positive pairs to $k = 10$. We set the the dimension of the hidden layer to $d' = 128$.

TABLE 2: Summary of the network reconstruction task on Cora ML, Citeseer and Cora networks for an embedding dimension of 25+25. For the computation of the t-statistic and corresponding p-value, we select the best benchmark algorithm (identified with an asterisk (*)) according to AP.

Network	Algorithm	Mean Rank	AUROC	AP	mAP	p@1	p@3	p@5	p@10
Cora ML	ATP (log)	4083.0(25.8)	0.515(0.003)	0.531(0.003)	0.044(0.001)	0.025(0.002)	0.042(0.002)	0.058(0.002)	0.085(0.006)
	LINE	4093.9(40.6)	0.514(0.005)	0.473(0.003)	0.060(0.002)	0.097(0.004)	0.110(0.004)	0.118(0.004)	0.124(0.007)
	$\text{G2G}_{\text{NA},K=1}$	54.4(4.9)	0.994(0.001)	0.991(0.001)	0.609(0.007)	0.551(0.014)	0.530(0.011)	0.517(0.012)	0.495(0.017)
	NK	101.7(3.7)	0.988(0.000)	0.987(0.001)	0.586(0.004)	0.607(0.007)	0.431(0.003)	0.339(0.002)	0.220(0.001)
	HNN	423.6(24.8)	0.95(0.003)	0.945(0.004)	0.22(0.01)	0.197(0.013)	0.159(0.008)	0.136(0.007)	0.106(0.005)
	HGCN	457(30.2)	0.946(0.004)	0.952(0.003)	0.42(0.013)	0.467(0.029)	0.318(0.01)	0.246(0.006)	0.162(0.004)
	$\text{G2G}_{K=1}$	64.6(7.7)	0.992(0.001)	0.989(0.001)	0.586(0.008)	0.536(0.013)	0.493(0.011)	0.479(0.013)	0.459(0.018)
	$\text{G2G}_{K=3}$	55.8(5.5)	0.993(0.001)	0.991(0.001)	0.609(0.011)	0.563(0.011)	0.539(0.014)	0.530(0.013)	0.509(0.013)
	$\text{HEADNet}_{\Sigma=\mathbb{I}}$	13.4(2.7)	0.999(0.000)	0.998(0.001)	0.792(0.003)	0.743(0.006)	0.553(0.003)	0.432(0.001)	0.270(0.001)
	$^*\text{G2G}_{\text{NA},K=3}$	44.0(5.6)	0.995(0.001)	0.992(0.001)	0.647(0.007)	0.588(0.011)	0.587(0.009)	0.575(0.010)	0.548(0.012)
	HEADNet	**4.0(1.4)**	**1.000(0.000)**	**0.999(0.000)**	**0.945(0.014)**	**0.937(0.016)**	**0.921(0.019)**	**0.912(0.021)**	**0.904(0.021)**
	p-value	7.09E-29	7.09E-29	1.09E-24	4.77E-53	4.89E-62	5.58E-50	3.62E-47	3.39E-51
Citeseer	ATP (log)	2171.6(19.9)	0.595(0.004)	0.601(0.006)	0.026(0.001)	0.008(0.001)	0.032(0.003)	0.039(0.004)	0.034(0.006)
	LINE	3156.9(29.5)	0.411(0.006)	0.418(0.002)	0.052(0.002)	0.053(0.002)	0.057(0.004)	0.050(0.004)	0.010(0.016)
	$\text{G2G}_{\text{NA},K=1}$	6.6(1.5)	0.999(0.000)	0.999(0.000)	0.814(0.009)	0.734(0.015)	0.611(0.012)	0.584(0.017)	0.570(0.066)
	NK	26.1(1.9)	0.995(0.000)	0.995(0.001)	0.722(0.004)	0.645(0.007)	0.365(0.003)	0.252(0.001)	0.141(0.000)
	HNN	840.1(122.7)	0.843(0.023)	0.854(0.018)	0.133(0.018)	0.093(0.016)	0.063(0.011)	0.051(0.008)	0.037(0.004)
	HGCN	873.6(129.9)	0.837(0.024)	0.868(0.023)	0.312(0.033)	0.31(0.029)	0.159(0.019)	0.11(0.013)	0.065(0.008)
	$\text{G2G}_{K=1}$	17.2(3.2)	0.997(0.001)	0.993(0.002)	0.632(0.010)	0.530(0.014)	0.395(0.014)	0.353(0.020)	0.424(0.069)
	$\text{G2G}_{K=3}$	17.0(2.8)	0.997(0.001)	0.993(0.002)	0.622(0.009)	0.513(0.014)	0.411(0.016)	0.371(0.023)	0.473(0.066)
	$\text{HEADNet}_{\Sigma=\mathbb{I}}$	13.6(1.8)	0.998(0.000)	0.994(0.002)	0.670(0.003)	0.571(0.005)	0.358(0.002)	0.253(0.001)	0.144(0.000)
	$^*\text{G2G}_{\text{NA},K=3}$	6.1(1.8)	0.999(0.000)	0.999(0.001)	0.790(0.010)	0.686(0.017)	0.634(0.016)	0.612(0.019)	0.634(0.054)
	HEADNet	7.3(2.1)	0.999(0.000)	0.996(0.002)	0.779(0.019)	**0.718(0.028)**	**0.709(0.024)**	**0.685(0.023)**	**0.711(0.084)**
	p-value	9.90E-01	9.91E-01	1.00E+00	9.95E-01	1.14E-06	2.22E-19	5.94E-19	5.64E-05
Cora	ATP (log)	33041.2(69.9)	0.494(0.001)	0.508(0.001)	0.049(0.000)	0.043(0.001)	0.053(0.001)	0.061(0.001)	0.077(0.002)
	LINE	32541.2(108.1)	0.502(0.002)	0.469(0.001)	0.087(0.001)	0.151(0.002)	0.157(0.002)	0.162(0.002)	0.166(0.003)
	$\text{G2G}_{\text{NA},K=1}$	135.6(11.0)	0.998(0.000)	0.997(0.000)	0.808(0.010)	0.795(0.011)	0.742(0.014)	0.724(0.013)	0.719(0.011)
	NK	401.7(11.3)	0.994(0.000)	0.993(0.000)	0.740(0.001)	0.787(0.002)	0.574(0.001)	0.449(0.001)	0.286(0.000)
	HNN	4958.1(174.1)	0.924(0.032)	0.921(0.013)	0.233(0.009)	0.232(0.007)	0.182(0.006)	0.157(0.004)	0.121(0.000)
	HGCN	1280.9(248.9)	0.98(0.046)	0.979(0.027)	0.493(0.02)	0.523(0.015)	0.381(0.012)	0.308(0.008)	0.212(0)
	$\text{G2G}_{K=1}$	181.7(20.9)	0.997(0.000)	0.996(0.000)	0.763(0.015)	0.746(0.017)	0.684(0.019)	0.672(0.018)	0.675(0.015)
	$\text{G2G}_{K=3}$	122.4(14.2)	0.998(0.000)	0.998(0.000)	0.813(0.013)	0.800(0.014)	0.756(0.016)	0.739(0.015)	0.735(0.012)
	$\text{HEADNet}_{\Sigma=\mathbb{I}}$	25.7(2.7)	1.000(0.000)	0.999(0.000)	0.929(0.001)	0.919(0.002)	0.694(0.001)	0.544(0.001)	0.337(0.000)
	$^*\text{G2G}_{\text{NA},K=3}$	81.3(10.1)	0.999(0.000)	0.998(0.000)	0.865(0.008)	0.854(0.008)	0.824(0.011)	0.802(0.011)	0.783(0.009)
	HEADNet	**11.9(2.0)**	**1.000(0.000)**	**1.000(0.000)**	**0.973(0.003)**	**0.972(0.003)**	**0.966(0.004)**	**0.961(0.004)**	**0.956(0.005)**
	p-value	1.06E-27	1.06E-27	2.46E-25	1.39E-44	3.91E-46	4.39E-39	3.36E-42	2.33E-52

4.3 Network Reconstruction

We use network reconstruction (NR) to evaluate the capacity of the learned embeddings to reflect the original data [13]. We evaluate NR in the standard fashion [13]: That is, we first train a model to convergence using complete information. For NR, we then randomly select a set of true-negative ordered node pairs equal the to size of true-positive node pairs ($|E|$). Area-under-receiver operating characteristic (AUROC), and average precision (AP) are pair-based metrics, computed by ranking pairs from both sets inversely by distance with respect to the metric of the model. AUROC and AP are then readily computed from this ranked list of labelled pairs. 'Mean Rank' is the expected position of a true positive node pair in the set of selected negative pairs when sorting by distance. Mean average precision (mAP) and precision at k (p@k) are subtly different in that they are computed with respect to each node individually and then the mean over all nodes is reported. Firstly, ground-truth (positive) node pairs of the form (u, v) are grouped by source node u. For each u, mAP is computed for the true neighbours of u and a random sample of 1000 non-neighbours. For p@k, the top k closest nodes to each source node u are determined and we compute the expectation that a node in that set is a true neighbour. Each experiment is repeated 30 times and we report the mean and s.d..

Table 2 provides a summary of the network reconstruction results for embedding dimension 25+25. Comparing between like algorithms with and without attributes, for example: $\text{G2G}_{\text{NA},K=1}$ and $\text{G2G}_{K=1}$, we see that algorithms that do not use attributes outperform their equivalents with attributes. This suggests that attributes act as a natural form of regularisation on the learning process, preventing over-fitting to the given training data (in this case, trading perfect network reconstruction for better link prediction performance, see section 4.4). Note that, the selected benchmark algorithm in all three cases ($\text{G2G}_{\text{NA},K=3}$) does not use attributes in the learning process, and so lacks this regularisation. This leads to its poorer performance in the link prediction task (table 3) in three out of four networks with attributes. HEADNet, on the other hand, does use attributes in the learning process and is able to maintain competitive performance with $\text{G2G}_{\text{NA},K=3}$ for NR, while handily our-performing in the link prediction task. Similarly, by considering only benchmark algorithms that leverage attributes (HNN, HGCN, $\text{G2G}_{K=1}$, and $\text{G2G}_{K=3}$), HEADNet achieves superior performance. Comparison between HEADNet and $\text{HEADNet}_{\Sigma=\mathbb{I}}$ reveals that learning the variance matrix results in superior performance across all networks, compared with fixing it to the identity matrix. We further investigated this in section 4.7.3.

TABLE 3: Summary of the link prediction task on synthetic, Cora ML, Citeseer, Pubmed, Cora and Wiki Vote networks for embedding dimension 25+25. For each network, for the computation of the t-statistic and corresponding p-value, we select the best benchmark algorithm (identified with an asterisk (*)) according to AP.

Synthetic	**Mean Rank**	**AUROC**	**AP**	**mAP**	**Wiki Vote**	**Mean Rank**	**AUROC**	**AP**	**mAP**
ATP (log)	88.7(8.1)	0.493(0.041)	0.574(0.040)	0.017(0.020)	ATP (log)	5768.5(37.5)	0.444(0.004)	0.471(0.003)	0.013(0.001)
LINE	122.0(8.2)	0.301(0.037)	0.383(0.012)	0.018(0.009)	LINE	7171.1(48.1)	0.309(0.005)	0.376(0.002)	0.042(0.002)
$\text{G2G}_{\text{NA},K=1}$	47.0(6.7)	0.733(0.045)	0.651(0.051)	0.010(0.005)	$\text{G2G}_{\text{NA},K=1}$	478.7(32.7)	0.954(0.003)	0.941(0.004)	0.109(0.005)
$\text{G2G}_{\text{NA},K=3}$	35.2(6.3)	0.801(0.040)	0.738(0.049)	0.023(0.010)	$\text{G2G}_{\text{NA},K=3}$	476.3(33.7)	0.954(0.003)	0.932(0.006)	0.114(0.004)
*NK	34.6(9.8)	0.804(0.063)	0.738(0.072)	0.030(0.008)	*NK	199.1(7.8)	0.981(0.001)	0.979(0.001)	0.179(0.004)
$\text{HEADNet}_{\text{NA}}$	**22.3(4.0)**	**0.876(0.026)**	**0.866(0.032)**	**0.051(0.009)**	$\text{HEADNet}_{\text{NA}}$	**121.6(8.6)**	**0.988(0.001)**	**0.984(0.001)**	**0.218(0.004)**
p-value	7.68E-08	4.53E-07	1.95E-11	1.84E-13	p-value	7.88E-42	7.88E-42	3.25E-23	9.48E-41
Cora ML	**Mean Rank**	**AUROC**	**AP**	**mAP**	**Cora**	**Mean Rank**	**AUROC**	**AP**	**mAP**
ATP (log)	500.4(11.9)	0.407(0.014)	0.446(0.009)	0.031(0.004)	ATP (log)	3945.8(35.1)	0.396(0.005)	0.435(0.004)	0.038(0.001)
LINE	405.2(11.8)	0.520(0.014)	0.475(0.008)	0.063(0.008)	LINE	3159.6(36.1)	0.516(0.006)	0.473(0.003)	0.084(0.004)
$\text{G2G}_{\text{NA},K=1}$	27.1(3.6)	0.969(0.004)	0.965(0.005)	0.240(0.013)	$\text{G2G}_{\text{NA},K=1}$	74.8(6.6)	0.989(0.001)	0.989(0.001)	0.456(0.012)
$\text{G2G}_{\text{NA},K=3}$	26.6(3.6)	0.970(0.004)	0.966(0.006)	0.244(0.014)	$\text{G2G}_{\text{NA},K=3}$	70.9(7.5)	0.989(0.001)	0.990(0.001)	0.485(0.012)
NK	29.6(3.7)	0.966(0.004)	0.965(0.005)	0.272(0.015)	NK	109.4(7.5)	0.983(0.001)	0.984(0.001)	0.479(0.005)
$\text{G2G}_{K=3}$	24.2(3.0)	0.972(0.004)	0.967(0.006)	0.244(0.014)	$\text{G2G}_{K=1}$	66.5(5.3)	0.990(0.001)	0.989(0.001)	0.426(0.012)
*$\text{G2G}_{K=1}$	23.8(2.9)	0.973(0.003)	0.967(0.006)	0.241(0.012)	*$\text{G2G}_{K=3}$	64.1(5.2)	0.990(0.001)	0.990(0.001)	0.445(0.012)
$\text{HEADNet}_{\text{NA}}$	24.4(3.9)	0.972(0.005)	**0.974(0.004)**	**0.394(0.016)**	$\text{HEADNet}_{\text{NA}}$	86.6(5.9)	0.987(0.001)	0.989(0.001)	**0.623(0.008)**
p-value	7.22E-01	7.22E-01	2.69E-06	3.16E-43	p-value	1.00E+00	1.00E+00	1.00E+00	2.48E-51
HEADNet	**16.9(2.6)**	**0.981(0.003)**	**0.980(0.004)**	**0.415(0.017)**	HEADNet	**46.1(4.3)**	**0.993(0.001)**	**0.993(0.001)**	**0.661(0.006)**
p-value	5.41E-14	5.41E-14	2.59E-14	2.98E-43	p-value	3.01E-21	3.01E-21	2.35E-21	3.92E-50
Citeseer	**Mean Rank**	**AUROC**	**AP**	**mAP**	**Pubmed**	**Mean Rank**	**AUROC**	**AP**	**mAP**
ATP (log)	303.9(11.2)	0.435(0.021)	0.494(0.021)	0.029(0.004)	ATP (log)	7692.6(24.3)	0.132(0.003)	0.327(0.001)	0.083(0.005)
LINE	299.9(7.3)	0.442(0.014)	0.433(0.007)	0.040(0.005)	LINE	4859.8(52.3)	0.452(0.006)	0.433(0.003)	0.045(0.002)
$\text{G2G}_{\text{NA},K=1}$	28.8(4.6)	0.948(0.008)	0.962(0.007)	0.279(0.015)	$\text{G2G}_{\text{NA},K=1}$	66.0(7.5)	0.993(0.001)	0.991(0.001)	0.503(0.012)
$\text{G2G}_{\text{NA},K=3}$	28.5(4.6)	0.949(0.009)	0.962(0.007)	0.264(0.013)	NK	107.6(6.2)	0.988(0.001)	0.987(0.001)	0.475(0.005)
NK	39.1(6.2)	0.929(0.011)	0.945(0.007)	0.265(0.012)	$\text{G2G}_{K=1}$	284.1(19.9)	0.968(0.002)	0.958(0.003)	0.146(0.006)
$\text{G2G}_{K=1}$	15.0(3.0)	0.974(0.006)	0.972(0.009)	0.259(0.012)	$\text{G2G}_{K=3}$	229.6(10.4)	0.974(0.001)	0.965(0.002)	0.170(0.004)
*$\text{G2G}_{K=3}$	14.9(2.8)	0.974(0.005)	0.972(0.008)	0.247(0.012)	*$\text{G2G}_{\text{NA},K=3}$	39.2(4.9)	0.996(0.001)	0.995(0.001)	0.612(0.011)
$\text{HEADNet}_{\text{NA}}$	35.0(5.2)	0.937(0.010)	0.951(0.007)	**0.370(0.026)**	$\text{HEADNet}_{\text{NA}}$	**36.0(4.3)**	**0.996(0.000)**	**0.996(0.001)**	**0.781(0.009)**
p-value	1.00E+00	1.00E+00	1.00E+00	1.18E-25	p-value	4.68E-03	4.68E-03	2.10E-04	1.06E-53
HEADNet	**12.8(2.1)**	**0.978(0.004)**	0.975(0.007)	**0.414(0.020)**	HEADNet	73.4(6.8)	0.992(0.001)	0.989(0.001)	0.421(0.007)
p-value	1.21E-03	1.21E-03	9.95E-02	6.18E-39	p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00

Note that the purpose of NR is to preserve the training data – the structure observed by the model at the time of training. NR is typically evaluated in alongside link prediction (LP), which evaluates the ability of a model to predict new emerging links (see section 4.4), to measure the overall generalisation performance of the model. This is because there is often a trade-off between NR and LP, where a model can 'overfit' on the observed data resulting in high NR performance, but can not generalise well and predict new links, resulting in poorer LP performance. A high-quality embedder model should simultaneously do both.

TABLE 4: Link prediction on unseen nodes.

	Cora ML (300 nodes removed)			Pubmed (1972 nodes removed)		
	Mean Rank	**AUROC**	**AP**	**Mean Rank**	**AUROC**	**AP**
*$\text{G2G}_{K=1}$	417.1(67.1)	0.746(0.028)	0.690(0.029)	2771.0(231.7)	0.836(0.012)	0.811(0.013)
$\text{G2G}_{K=3}$	644.3(85.2)	0.605(0.042)	0.556(0.036)	4033.5(421.9)	0.761(0.022)	0.742(0.021)
HEADNet	**181.1(40.9)**	**0.890(0.018)**	**0.899(0.017)**	**1673.4(96.5)**	**0.901(0.004)**	**0.902(0.004)**
p-value	1.25E-21	6.78E-29	2.63E-34	1.10E-26	2.45E-29	5.56E-33
	Citeseer (423 nodes removed)			Cora (1980 nodes removed)		
	Mean Rank	**AUROC**	**AP**	**Mean Rank**	**AUROC**	**AP**
*$\text{G2G}_{K=1}$	229.8(25.2)	0.772(0.018)	0.789(0.019)	2391.3(257.9)	0.807(0.018)	0.772(0.020)
$\text{G2G}_{K=3}$	233.9(29.3)	0.768(0.023)	0.785(0.024)	3522.5(334.9)	0.716(0.023)	0.668(0.027)
HEADNet	**154.5(21.0)**	**0.847(0.018)**	**0.864(0.016)**	**549.4(48.7)**	**0.956(0.003)**	**0.959(0.002)**
p-value	2.68E-18	3.80E-23	1.76E-23	4.62E-28	5.12E-30	1.09E-30

4.4 Link Prediction

To evaluate LP ability, we randomly select 10% of the edges in the network and remove them (ensuring that every node has at least one connecting edge) [7]. We randomly select also an equal number of randomly selected non-edges in the network. We then train each model on the incomplete network and rank the pairs of nodes based on distance.

Table 3 provides a summary of the LP results. We observe that, in general, HEADNet performs very well compared with the other benchmark algorithms on all network datasets. Furthermore, incorporating node attributes in the learning process improves the performance on all three algorithms that are capable of doing so on three out of the four networks: $\text{G2G}_{K=1}$ vs. $\text{G2G}_{\text{NA},K=1}$, $\text{G2G}_{K=3}$ vs. $\text{G2G}_{\text{NA},K=3}$, and HEADNet vs. $\text{HEADNet}_{\text{NA}}$. The poorer performance on the Pubmed network perhaps suggests that the *homophily* property (nodes with similar attributes are more likely to connect) does not hold for this network. Still, the performance of HEADNet for this network (and all others) is notably superior to other node-attribute conscious benchmark algorithms.

4.4.1 Unseen Nodes

Mapping directly from attributes allows our model to handle previously unseen nodes. To evaluate the capacity for HEADNet to predict edges on previously unseen nodes, we perform the following: We randomly sample 10% nodes from the network and remove all in- and out- going edges from each of these selected nodes (this may result in the

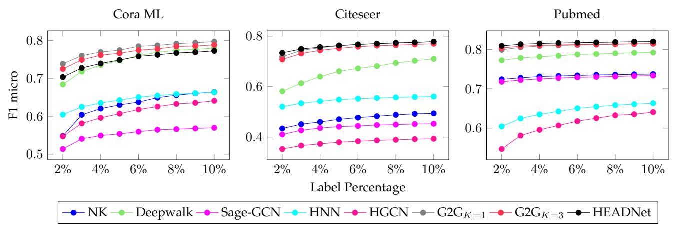


Fig. 3: Mean of micro-averaged F1 scores for node classification on the Cora ML, Citeseer and Pubmed networks for embedding dimension 5 and a range of observed label percentages.

removal of additional nodes if we remove all of a nodes neighbours). For each removed edge, we randomly select a pair of nodes as a negative sample. We then train and evaluate in the same way as in the link prediction experiment. We repeat this process 30 times for each network. We compare only against G2G as only G2G can handle unseen nodes.

Table 4 provides a summary of the results of this experiment for embedding dimension 5+5. From this, we see that HEADNet can achieve substantially better results than G2G – outperforming it on all networks by all metrics – in relatively small embedding dimensions. These results suggest that embedding to a hyperbolic metric space using attributes can provide an effective method for predicting the links of previously unseen nodes.

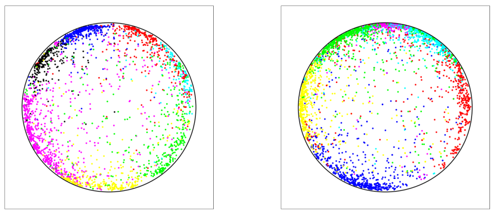
4.5 Node Classification

A popular downstream task for low-dimensional node representation is node classification (NC). Often in complex systems, entities belong to one to many classes that can affect the emergence of links or new attributes. To evaluate NC, we learn low-dimensional embeddings using the full network structure in an unsupervised manner. We then use a Support Vector Classifier (SVC) to evaluate the separation of classes in the embedding space For hyperbolic embeddings, we first project to the Klein model of hyperbolic space, which preserves straight lines [54]. For HEADNet and G2G, we use the mean vectors of the learned distributions as the input features to the SVC. To simulate a common scenario, where only a small fraction of nodes are labelled, we train our SVC using small fractions of node labels, ranging from 2%-10%.

Figure 3 reports the mean micro-averaged F1 score achieved on the node classification task for a range a label percentages that are used to train the SVC. For two out of the three networks, we observe that HEADNet marginally out-performs all benchmark algorithms, even with only 2% labels observed (0.733 vs. 0.718 and 0.809 vs. 0.803 for the Citeseer and Pubmed networks respectively). For the Cora ML network, we see inferior performance compared with the best benchmark (for example, HEADNet achieves an F1 score of 0.748 at 5% observed labels, whereas G2G$_{K=1}$ scores 0.774). However, as we discuss in section 5.3, we can view this performance as a lower bound that can be improved in future work.

4.6 Network Visualisation

One common application of network embedding is network visualisation. Figure 4 illustrates two-dimensional Poincaré



(a) Cora ML (b) Citeseer

Fig. 4: Poincaré visualisations of the learned hyperboloid mean vectors for the nodes in the Cora ML and Citeseer networks. Nodes are coloured according to class labels, however class labels were not used in the learning process.

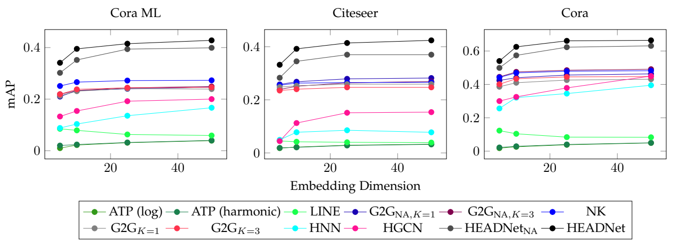


Fig. 5: mAP scores for link prediction on the Cora ML, Citeseer and Cora networks across a range of dimensions.

projections of the hyperboloid mean vectors learned by HEADNet in the 2+2 dimension setting for the Cora ML and Citeseer networks. Nodes are coloured according to node label. Note that HEADNet is a purely unsupervised technique, and is trained using network topology and node attributes only. The visualisation demonstrates that homogeneous nodes (i.e. nodes belonging to the same class) are clustered together in the low-dimensional space and this supports our intuition that consideration of node attributes, as well as network topology, can lead to superior (and more robust) performance in downstream tasks.

4.7 Parameter Sensitivity

4.7.1 *Robustness to Choice of Embedding Dimension*

We evaluated the robustness of HEADNet to choice of embedding dimension, measuring mAP achieved in the LP task for embedding dimensions 2+2, 5+5, 10+10, 25+25, and 50+50. Figure 5 summarises these results for the for the Cora ML, Citeseer and Cora networks. Te observe the robustness of HEADNet to the setting of embedding dimension, even at low dimensions, as well as a consistent performance versus the benchmark algorithms.

4.7.2 *Choice of Metric*

In order investigate the impact of the metric function on the performance of downstream tasks, we compare the performance of HEADNet using a hyperboloid metric and a Euclidean one. To achieve this, we omit applying Exp_{μ_0} and Ψ_u and apply eq. (7) directly on the output of $\mathbf{W}_{\mathbb{H}^n}\mathbf{h}_u + \mathbf{b}_{\mathbb{H}^n}$.

Table 5 compares the performance of using a Euclidean metric versus a hyperbolic metric on a number of tasks across a range of embedding dimensions. Based on the mAP

TABLE 5: The effect of a Euclidean ($\mathbb{R}$) versus Hyperbolic ($\mathbb{H}$) metric on three downstream tasks: NR, LP and NC, over a range of embedding dimensions n.

		NR (mAP)		LP (mAP)		NC (10 fold F1)	
n	Network	$\mathbb{R}$	$\mathbb{H}$	$\mathbb{R}$	$\mathbb{H}$	$\mathbb{R}$	$\mathbb{H}$
5	Cora ML	0.389(0.012)	0.833(0.018)	0.229(0.050)	0.341(0.022)	0.833(0.015)	0.800(0.034)
	Citeseer	0.387(0.008)	0.675(0.021)	0.245(0.015)	0.332(0.023)	0.808(0.005)	0.793(0.003)
	Pubmed	0.334(0.041)	0.449(0.014)	0.256(0.049)	0.302(0.007)	0.832(0.039)	0.805(0.049)
	Cora	0.605(0.045)	0.890(0.007)	0.430(0.002)	0.540(0.010)	0.482(0.045)	0.439(0.023)
10	Cora ML	0.604(0.024)	0.926(0.019)	0.272(0.011)	0.395(0.020)	0.856(0.039)	0.843(0.027)
	Citeseer	0.433(0.033)	0.752(0.025)	0.241(0.037)	0.392(0.026)	0.819(0.044)	0.797(0.004)
	Pubmed	0.416(0.040)	0.544(0.011)	0.312(0.010)	0.374(0.008)	0.854(0.007)	0.831(0.005)
	Cora	0.818(0.026)	0.961(0.005)	0.482(0.014)	0.625(0.006)	0.606(0.030)	0.517(0.001)
25	Cora ML	0.647(0.040)	0.939(0.016)	0.296(0.001)	0.415(0.017)	0.861(0.039)	0.848(0.040)
	Citeseer	0.486(0.013)	0.773(0.022)	0.278(0.043)	0.414(0.020)	0.832(0.049)	0.800(0.031)
	Pubmed	0.506(0.007)	0.616(0.009)	0.359(0.025)	0.421(0.007)	0.857(0.044)	0.850(0.043)
	Cora	0.852(0.046)	0.973(0.003)	0.510(0.031)	0.661(0.006)	0.633(0.045)	0.585(0.045)
50	Cora ML	0.681(0.038)	0.945(0.014)	0.285(0.032)	0.428(0.023)	0.857(0.014)	0.858(0.011)
	Citeseer	0.485(0.003)	0.779(0.019)	0.284(0.034)	0.424(0.017)	0.846(0.037)	0.826(0.047)
	Pubmed	0.511(0.028)	0.641(0.007)	0.346(0.045)	0.436(0.005)	0.860(0.021)	0.855(0.039)
	Cora	0.843(0.016)	0.974(0.005)	0.538(0.038)	0.664(0.007)	0.634(0.020)	0.634(0.024)

TABLE 6: mAP scores achieved by HEADNet on the LP task with and without learning the variance matrix.

	Dimension	5	10	25	50
Cora ML	HEADNet$_{\Sigma=\mathbb{I}}$	0.277(0.019)	0.297(0.016)	0.302(0.015)	0.306(0.017)
	HEADNet	**0.341(0.022)**	**0.395(0.020)**	**0.415(0.017)**	**0.428(0.023)**
	p-value	1.41E-17	2.84E-28	2.36E-34	2.37E-30
Citeseer	HEADNet$_{\Sigma=\mathbb{I}}$	0.274(0.019)	0.291(0.016)	0.301(0.014)	0.311(0.017)
	HEADNet	**0.332(0.023)**	**0.392(0.026)**	**0.414(0.020)**	**0.424(0.017)**
	p-value	1.13E-15	4.23E-23	1.80E-31	1.55E-33
Pubmed	HEADNet$_{\Sigma=\mathbb{I}}$	0.317(0.007)	0.389(0.007)	0.436(0.008)	0.441(0.007)
	HEADNet	0.302(0.007)	0.374(0.008)	0.421(0.007)	0.436(0.005)
	p-value	1.00E+00	1.00E+00	1.00E+00	1.00E+00
Cora	HEADNet$_{\Sigma=\mathbb{I}}$	0.484(0.005)	0.558(0.006)	0.575(0.006)	0.578(0.008)
	HEADNet	**0.540(0.010)**	**0.625(0.006)**	**0.661(0.006)**	**0.664(0.007)**
	p-value	1.58E-29	1.58E-46	2.31E-46	2.31E-46

scores for for both NR and LP, we can see that the hyperbolic metric does indeed out-perform the Euclidean one handsomely, and this strongly suggests that the intuition behind our approach is sensible and the extra steps required to map between spaces is justified and worthwhile. However, the performance on the NC task further supports our theory (see section 4.5) that projecting to the Klein model and using a Euclidean classifier will achieve a lower-bound of performance versus a hyperbolic classifier.

4.7.3 The Effect of Learning a Variance Matrix

Based on our findings for the NR experiment (table 2), and in order to confirm our hypothesis that learning both mean and variance vectors would result in superior downstream performance, we run an ablation study to compare mAP on the link prediction task with and without learning Σ.

Table 6 provides a comparison of mAP scores achieved in the link prediction task by HEADNet with and without fixing the variance matrix to the identity matrix (HEADNet$_{\Sigma=\mathbb{I}}$). We find that in three out of the four citation networks, performance is significantly improved when both the mean and variance matrices are learned simultaneously, even a significance value of $1.58E-29$ for a 5-dimensional embedding of the Cora network. We also observe that this holds true across the range of tested dimensions.

5 Discussion

5.1 Connection to Variational Autoencoders

A natural comparison can be made between HEADNet and a Variational Autoencoder (VAE) [28]. VAEs bridge the gap between autoencoders and the generation of new, meaningful data by regularising the latent space representation of the data by encoding not to points in the latent space, but to distributions and applying a penalty to the distribution to encourage them to be as close to the chosen prior, $p(\mathbf{z})$, as possible. In this way, both a local (due to variance control) and global (due to mean control) regularisation of the latent space is ensured. The connection between VAEs and HEADNet stems from the commonly selected choice of prior: $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$, a standard Gaussian distribution; as well as the latent space penalty coming the form of Kullback Leibler divergence $D_{KL}(f(\mathbf{x}), \mathcal{N}(\mathbf{0}, \mathbb{I}))$. Like HEADNet, the encoder makes the common simplifying assumption of feature independence to encode data points to d-dimensional means and d-dimensional vectors corresponding to diagonal covariance matrices (i.e., $f(\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$). Furthermore, the architecture of the encoder neural network is similar to the architecture of the embedder for HEADNet [28]. Through this lens, we can view HEADNet as an encoder that learns uses not one, but N priors (corresponding to the N nodes in the network) for regularisation, and that the priors are not fixed (only the shape of them). We leverage the structure in the network to select priors that we use to regularise each distribution, based on the distributions of the neighbours in the network.

A key difference between a VAE and HEADNet is the omission of the decoder in HEADNet, and this leads to a natural extension for HEADNet for the future: by simultaneously learning a hyperbolic decoder along with our proposed embedder, we would be able to generate new, meaningful entities in our system. For example: consider a social network where nodes are annotated with characteristics that describe a person. Each link describes a friendship, and we find that the network displays characteristics that would make us believe that a hyperbolic metric space may underpin it. After training a hyperbolic encoder (HEADNet) and a decoder, we would be able to, given a person, "generate people" that might be friends with that person by sampling from the learned distribution for that person and decoding to obtain the attributes that that theoretical person may exhibit. We leave this extension as future work.

5.2 Undirected Network Embedding

Undirected and directed network embedding share a close relationship. Many undirected embedding algorithms can be adapted to represent directed relationships, namely by learned two representations of each node u – one to be used for outgoing edges $\mathbf{u}_o$ and and one for incoming $\mathbf{u}_i$. For two nodes $u, v \in V$, where $\mathbf{u}_o \neq \mathbf{u}_i$, then asymmetric similarities can be inferred, $(f(\mathbf{u}_o, \mathbf{v}_i) \neq f(\mathbf{v}_o, \mathbf{u}_i)$ for most common vector similarity measures f). However, in the worst case, this could require two models and therefore double the number of parameters. This is one of the drawbacks that parameter sharing overcomes. For HEADNet, we learn $2n$ parameters for n-dimensional node representations, but the first layer of both models is shared. Further, the use of two representations for a single node introduces the problem of downstream representation: should $\mathbf{u}_o$, $\mathbf{u}_i$ or some aggregation or concatenation of them be used for, for example, node classification? By learning embeddings of distributions, the mean vectors correspond the the 'most probable' location of the node in the low-dimensional space and so makes for an attractive vector to use in downstream tasks – exactly as we have done for the node classification and network visualisation tasks in section 4.5 and section 4.6 respectively. Of course, the relatedness of the tasks of undirected and directed embedding cannot be ignored. Indeed, by fixing $\Sigma = \mathbb{I}$ (and therefore ignoring direction), our loss function (eq. (12)) degenerates into the loss function of HEAT [27].

5.3 Node Classification

Sections 4.5 and 4.7.2 together suggest that mapping from the hyperboloid to the Klein model and using Klein coordinates directly as input into a Euclidean classifier may result in sub-optimal class separation for some networks. Despite preserved straight lines, it is challenging for the SVC to build satisfactory class boundaries because distances are distorted between Euclidean and Klein metrics and most nodes will be embedded far from the origin of the space, where large hyperbolic distances are represented by increasingly small Euclidean ones. This is especially apparent in the direct hyperboloid to Euclidean comparison in section 4.7.2, as the Euclidean version of HEADNet outperforms the hyperbolic version across the board.

To overcome this, one could account for the distortion as much as possible by first re-centring hyperbolic projections through a circle inversion about the Frècet mean [46] to centre the points about the origin and therefore maximise the Euclidean space between points in the Klein projection. Alternatively, one could fit a hyperbolic neural network classifier [20] to the mean vectors in order to make node class predictions. This will fully respect the hyperbolic metric space in learning decision boundaries, which we feel will result in the performance gain versus a Euclidean metric that we currently observe in NR and LP. Visual inspection of two-dimension network visualisation (see fig. 4) confirm that the classes appear to sensibly separated and – since we intended to focus primarily on directed links between nodes in this work, and since the NR and LP results are promising – we leave optimising the NC performance as future work.

5.4 Selecting Embedding Dimension

It has been shown in the past that a low-dimensional hyperbolic embedding is sufficient to achieve high mAP on tree and tree-like structures, such as taxonomies and dendrograms [23]. For example: De Sa et *al.* [12] achieved a near-perfect mAP score of 0.989 using just 2 dimensions on the Wordnet hypernym graph, a strict tree structure. Furthermore, it has been shown that the relative advantages of a hyperbolic versus Euclidean network embedding diminish as the dimensionality increases [27]. As such, we suggest measuring the power-law 'fit' of the network in question (for example, using powerlaw [55]), and selecting between 2-10 dimensions based on the fit. The small dimension size would maximise the efficacy over a Euclidean embedding, while being very quick to generate. One could also hold out a small proportion of edges from a network in question and generate a range of embeddings of dimension 2-10 (as table 4 shows – HEADNet can generalise well to predict the distributions of unseen nodes even in low dimensions), compute mAP and select the embedding based on the knee-point of the mAP versus dimension curve.

6 Conclusion

In this paper we have presented HEADNet to fill the gap of embedding attributed and directed networks in hyperbolic space. We propose an embedding model to map attributed nodes to Gaussian distributions in hyperbolic space. Based on previous works [49], we further propose the use of a node similarity measure in hyperbolic space based on a novel mapping procedure to map hyperbolic Gaussian distributions to Euclidean ones, preserving hyperbolic distance and direction. We achieve competitive performance on a number of downstream machine learning tasks, including predicting the links for previously unseen nodes. Our results suggest that HEADNet can provide a general inductive hyperbolic embedding method for directed networks with and without node attributes, opening the door to hyperbolic manifold learning on a wider range of network than previously possible.

References

[1] F. Papadopoulos, M. Kitsak, M. Serrano, M. Boguná, and D. Krioukov, "Popularity versus similarity in growing networks," *Nature*, vol. 489, 2011.

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[3] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[4] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[5] D. Krioukov, F. Papadopoulos, A. Vahdat, and M. Boguñá, "Curvature and temperature of complex networks," *Physical Review E*, vol. 80, no. 3, p. 035101, 2009.

[6] F. Papadopoulos, C. Psomas, and D. Krioukov, "Network mapping by replaying hyperbolic growth," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 1, pp. 198–211, 2015.

[7] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[8] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[9] G. Alanis-Lobato, P. Mier, and M. A. Andrade-Navarro, "Efficient embedding of complex networks to hyperbolic space via their laplacian," *Scientific Reports*, vol. 6, 2016.

[10] ——, "Manifold learning and maximum likelihood estimation for hyperbolic network embedding," *Applied Network Science*, vol. 1, no. 1, p. 10, 2016.

[11] R. Sarkar, "Low distortion delaunay embedding of trees in hyperbolic plane," in *International Symposium on Graph Drawing*. Springer, 2011, pp. 355–366.

[12] C. De Sa, A. Gu, C. Ré, and F. Sala, "Representation tradeoffs for hyperbolic embeddings," *Proceedings of machine learning research*, vol. 80, p. 4460, 2018.

[13] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems*, 2017, pp. 6338–6347.

[14] ——, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *International Conference on Machine Learning*, 2018, pp. 3776–3785.

[15] B. Dhingra, C. Shallue, M. Norouzi, A. Dai, and G. Dahl, "Embedding text in hyperbolic spaces," in *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, 2018, pp. 59–69.

[16] B. P. Chamberlain, J. Clough, and M. P. Deisenroth, "Neural embeddings of graphs in hyperbolic space," *arXiv:1705.10359*, 2017.

[17] M. Leimeister and B. J. Wilson, "Skip-gram word embeddings in hyperbolic space," *arXiv preprint arXiv:1809.01498*, 2018.
[18] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5337–5344.
[19] R. C. Wilson, E. R. Hancock, E. Pekalska, and R. P. Duin, "Spherical and hyperbolic embeddings of data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2255–2269, 2014.
[20] O.-E. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2018, pp. 5350–5360.
[21] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4869–4880.
[22] I. Balazevic, C. Allen, and T. Hospedales, "Multi-relational poincaré graph embeddings," *Advances in Neural Information Processing Systems*, vol. 32, pp. 4463–4473, 2019.
[23] C. Gulcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. M. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro *et al.*, "Hyperbolic attention networks," *arXiv preprint arXiv:1805.09786*, 2018.
[24] Z. Wu, Z. Di, and Y. Fan, "A hyperbolic embedding model for directed networks," *arXiv preprint arXiv:1906.03597*, 2019.
[25] O. Ganea, G. Becigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings," in *International Conference on Machine Learning*, 2018, pp. 1646–1655.
[26] R. Suzuki, R. Takahama, and S. Onoda, "Hyperbolic disk embeddings for directed acyclic graphs," in *International Conference on Machine Learning*, 2019, pp. 6066–6075.
[27] D. McDonald and S. He, "Heat: Hyperbolic embedding of attributed networks," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2020, pp. 28–40.
[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[29] E. Mathieu, C. Le Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh, "Continuous hierarchical representations with poincaré variational auto-encoders," in *Advances in neural information processing systems*, 2019, pp. 12 565–12 576.
[30] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2016, pp. 2014–2023.
[31] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.
[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.
[33] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geomgcn: Geometric graph convolutional networks," *arXiv preprint arXiv:2002.05287*, 2020.
[34] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
[35] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKDD*. ACM, 2014, pp. 701–710.
[36] C. Hou, S. He, and K. Tang, "RoSANE: Robust and scalable attributed network embedding for sparse networks," *Neurocomputing*, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.05.080
[37] C. Hou, H. Zhang, S. He, and K. Tang, "Glodyne: Global topology preserving dynamic network embedding," *Accepted by IEEE Transactions on Knowledge and Data Engineering*, 2020.
[38] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.
[39] J. Gibert, E. Valveny, and H. Bunke, "Graph embedding in vector spaces by node attribute statistics," *Pattern Recognition*, vol. 45, no. 9, pp. 3072–3083, 2012.
[40] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-first AAAI conference on artificial intelligence*, 2017.
[41] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 387–396.
[42] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD*. ACM, 2016, pp. 855–864.
[43] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.
[44] J. Sun, B. Bandyopadhyay, A. Bashizade, J. Liang, P. Sadayappan, and S. Parthasarathy, "Atp: Directed graph embedding with asymmetric transitivity preservation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 265–272.
[45] W. F. Reynolds, "Hyperbolic geometry on a hyperboloid," *The American mathematical monthly*, vol. 100, no. 5, pp. 442–455, 1993.
[46] B. Wilson and M. Leimeister, "Gradient descent in hyperbolic space," *arXiv:1805.08207*, 2018.
[47] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.
[48] J. R. Clough and T. S. Evans, "Embedding graphs in lorentzian spacetime," *PloS one*, vol. 12, no. 11, p. e0187301, 2017.
[49] Y. Nagano, S. Yamaguchi, Y. Fujita, and M. Koyama, "A wrapped normal distribution on hyperbolic space for gradient-based learning," in *International Conference on Machine Learning*, 2019, pp. 4693–4702.
[50] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.
[51] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2003, pp. 132–139.
[52] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.
[53] J. Sun, D. Ajwani, P. K. Nicholson, A. Sala, and S. Parthasarathy, "Breaking cycles in noisy hierarchies," in *Proceedings of the 2017 ACM on Web Science Conference*, ser. WebSci '17. New York, NY, USA: ACM, 2017, pp. 151–160. [Online]. Available: http://doi.acm.org/10.1145/3091478.3091495
[54] J. W. Cannon, W. J. Floyd, R. Kenyon, W. R. Parry *et al.*, "Hyperbolic geometry," *Flavors of geometry*, vol. 31, pp. 59–115, 1997.
[55] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: a python package for analysis of heavy-tailed distributions," *PloS one*, vol. 9, no. 1, p. e85777, 2014.

David McDonald was born in the United Kingdom in 1993. He received the B.Sc., M.Sc., and Ph.D. degrees from Birmingham University, West Midlands, United Kingdom, in 2015, 2016, and 2020, respectively. He now works for AIA Insights Ltd. as a Chief Technical Officer. His main research interests are complex networks, network embedding, heuristic searches and machine learning.

Shan He is a Senior Lecturer (Tenured Associate Professor) in School of Computer Science, the University of Birmingham. He is also an affiliate of the Centre for Computational Biology. His research interests include complex networks, machine learning, optimisation, and their applications to biomedicine. Shan is an Associate Editor of IEEE Transactions on Nanobioscience and Complex & Intelligent Systems (Springer).