Broader Impacts

This article introduce a method which can improve the performance on graph neural networks. We do not believe there are any foreseeable negative societal impacts beyond generic points that apply to any graph learning method.

Appendix Outline

Section A summarizes the notation and objects used in this paper. Section B gives general background on tensor and the algebra H. Section C discusses the feature map $\widetilde{\varphi}$ for sequences and possible choices for the lift φ from labels to the algebra H. Section D contains the proofs of our main theorems and some variations of hypoelliptic diffusion. Section E provides the formal statement of Theorem 2 and the extension to pooled features. Section F gives background and details of the low-rank algorithm. Section G discusses variations of the sequence feature map which lead to different node and graph features. Section H includes further experiments and discussion on the empirical results.

A Notation

Symbol	Meaning
	Fixed Parameters and Indices
d	dimension of node attributes
k	length of random walk
n	number of nodes in graph
	Sequence Features
$\mathbb{R}^d$	finite dimensional vector space for node attributes
$\mathsf{Seq}(\mathbb{R}^d)$	sequences $\mathbf{x} = (x_0, \ldots, x_k)$ of arbitrary length k in $\mathbb{R}^d$
$\delta_k \mathbf{x}$	increments of a sequence where $\delta_0 \mathbf{x} := x_0$ and $\delta_k \mathbf{x} := x_k - x_{k-1}$ for $k > 0$
H	tensor algebra of $\mathbb{R}^d$ (see Appendix B)
φ	algebra lifting $\varphi : \mathbb{R}^d \to H$
$\exp_\otimes$	tensor exponential $\exp_\otimes : \mathbb{R}^d \to H$ (main example of algebra lifting)
$\widetilde{\varphi}$	sequence feature map $\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H$, where $\widetilde{\varphi}(\mathbf{x}) = \varphi(\delta_0 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x})$
	Graphs, Adjacency and Laplacian Matrices
$\mathcal{G}$	$\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ graph with vertex set, edge set, and node attributes $f : \mathcal{V} \to \mathbb{R}^d$
$(B_k)_{k \geq 0}$	simple random walk on graph (valued in $\mathcal{V}$)
$(L_k)_{k \geq 0}$	lifted random walk over $\mathbb{R}^d$, $L_k := f(B_k)$
$\mathbf{L}_k$	lifted length k random walk $\mathbf{L}_k = (L_0, \ldots, L_k)$
Φ	$\Phi(i) = \mathbb{E}[\widetilde{\varphi}(L_1, \ldots, L_n) \mid L_0 = i] \in H$ feature map for node $i \in \mathcal{V}$
Ψ	$\Psi(\mathcal{G}) = \mathbb{E}[\widetilde{\varphi}(L_1, \ldots, L_n)] \in H$ feature map for graphs
A	standard adjacency matrix
$\widetilde{A}$	tensor adjacency matrix
P	standard transition matrix
$\widetilde{P}$	tensor transition matrix
$\mathcal{L}$	normalized graph Laplacian
$\widetilde{\mathcal{L}}$	normalized hypo-elliptic graph Laplacian

Notation Conventions

- Bold symbols are used for tensors and sequences; vectors are unbolded, such as $x \in \mathbb{R}^d$
- Coordinates for vectors are denoted using superscripts: $x = (x^{(1)}, \dots, x^{(d)}) \in \mathbb{R}^d$.
- Tensors are denoted by $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots) \in H$, where $\mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}$.
- Sequences are denoted by $\mathbf{x} = (x_0, \dots, x_k) \in \mathsf{Seq}(\mathbb{R}^d)$.

B Tensors and the Algebra H

In this section, we provide a brief overview of tensors on a finite-dimensional vector space $\mathbb{R}^d$, along with the resulting algebra H.

Tensors on $\mathbb{R}^d$. While tensor products between vector spaces are defined more generally, our main focus is on defining the tensor powers of $\mathbb{R}^d$, denoted by $(\mathbb{R}^d)^{\otimes m}$ for some $m \in \mathbb{N}$. The tensor power $(\mathbb{R}^d)^{\otimes m}$ is also a vector space. Given a basis $e^1, \dots, e^d$ of $\mathbb{R}^d$, we can define a basis of $(\mathbb{R}^d)^{\otimes m}$ as the collection of all

$$e^I := e^{i_1} \otimes \ldots \otimes e^{i_m}$$

over all *multi-indices* $(i_1, \dots, i_m)$, where each $i_j \in [d]$. Thus, any element $\mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}$ can be represented as

$$\mathbf{v}_m = \sum_{i_1, \dots, i_m = 1}^{d} \mathbf{v}_m^{(i_1, \dots, i_m)} e^{(i_1, \dots, i_m)},$$

where the $\mathbf{v}_m^{(i_1, \dots, i_m)} \in \mathbb{R}$ specifies the coordinates of $\mathbf{v}_m$. In particular, $(\mathbb{R}^d)^{\otimes 1}$ is simply $\mathbb{R}^d$ itself; $(\mathbb{R}^d)^{\otimes 2}$ can be viewed as the space of $d \times d$ matrices; $(\mathbb{R}^d)^{\otimes 3}$ is the space of $d \times d \times d$ arrays, etc.

Given two vectors $x, y \in \mathbb{R}^d$ which we represent coordinate-wise as

$$x = (x^{(1)}, \dots, x^{(d)}), \qquad y = (y^{(1)}, \dots, y^{(d)}),$$

the tensor product $x \otimes y \in (\mathbb{R}^d)^{\otimes 2}$ is given by

$$(x \otimes y)^{(i,j)} := x^{(i)} y^{(j)}.$$

More generally, if $\mathbf{u}_m \in (\mathbb{R}^d)^{\otimes m}$ and $\mathbf{v}_n \in (\mathbb{R}^d)^{\otimes n}$, the tensor product $\mathbf{u}_m \otimes \mathbf{v}_n \in (\mathbb{R}^d)^{\otimes (m+n)}$ is defined coordinate-wise by

$$(\mathbf{u}_m \otimes \mathbf{v}_n)^{(i_1, \dots, i_{m+n})} := \mathbf{u}_m^{(i_1, \dots, i_m)} \mathbf{v}_n^{(i_{m+1}, \dots, i_{m+n})}.$$

Furthermore, we note that $(\mathbb{R}^d)^{\otimes m}$ inherits an inner product from $\mathbb{R}^d$ through a choice of basis, as is shown in Equation (3).

Free Algebras. Our goal is to find a vector space H that contains the vector space $\mathbb{R}^d$ but is large enough to support a richer algebraic structure; in particular, a multiplication. Formally, this means we look for an injective map $\mathbb{R}^d \hookrightarrow H$ into an algebra H. A classic mathematical construction that turns a vector space into an algebra is the *free associative algebra*, defined as

$$\bigoplus_{m=0}^{\infty} (\mathbb{R}^d)^{\otimes m} = \{(\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_M, 0, \dots) \,:\, \mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m},\, M \in \mathbb{N}\},$$

where addition and multiplication of two elements $\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \dots)$ and $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \dots)$ is given by defining the $(\mathbb{R}^d)^{\otimes m}$ coordinate to be

$$(\mathbf{u} + \mathbf{v})_m = \mathbf{u}_m + \mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}, \qquad (\mathbf{u} \cdot \mathbf{v})_m = \sum_{i=0}^{m} \mathbf{v}_i \otimes \mathbf{w}_{m-i} \in (\mathbb{R}^d)^{\otimes m}.$$

We emphasize that the multiplication is not commutative.

A Universal Property. Indeed, this construction is the most general way to turn $\mathbb{R}^d$ into an algebra as the following classical result shows: if we denote with $\iota : \mathbb{R}^d \hookrightarrow \bigoplus_{m=0}^{\infty}(\mathbb{R}^d)^{\otimes m}$ the embedding $\iota(x) = (0, x, 0, \dots)$ then any linear map from $\mathbb{R}^d$ into any associative algebra A factors through ι. In other words, given an associative algebra A and any linear map $h : \mathbb{R}^d \to A$, there exists a homomorphism of algebras $\tilde{h} : \bigoplus_{m=0}^{\infty}(\mathbb{R}^d)^{\otimes m} \to H$ such the following diagram commutes

$$\begin{tikzcd}
\mathbb{R}^d \arrow[r, "h"] \arrow[d, "\iota"'] & H \\
\bigoplus_{m=0}^{\infty}(\mathbb{R}^d)^{\otimes m} \arrow[ur, dashed, "\tilde{h}"'] & .
\end{tikzcd}$$

In a sense, this shows that $\bigoplus_{m=0}^{\infty}(\mathbb{R}^d)^{\otimes m}$ is the "most general algebra" that $\mathbb{R}^d$ embeds into.

An Inner Product. While the free associative algebra satisfies the above universal property, it is convenient to work in the slightly larger space

$$\prod_{m\geq 0}(\mathbb{R}^d)^{\otimes m} = \{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots) : \mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}\}$$

and define an inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{m=0}^{\infty} \langle \mathbf{u}_m, \mathbf{v}_m \rangle_m \tag{16}$$

for elements $\mathbf{u}, \mathbf{v} \in \prod_{m\geq 0}(\mathbb{R}^d)^{\otimes m}$ for which the sum (16) converges (it does not converge for general elements of $\prod_{m\geq 0}(\mathbb{R}^d)^{\otimes m}$). The largest space for which this inner product exists, is

$$H := \{\mathbf{v} \in \prod_{m\geq 0}(\mathbb{R}^d)^{\otimes m} \ : \ \|\mathbf{v}\| < \infty\} \text{ and } \|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}.$$

The space H contains all the properties we require; it contains $\mathbb{R}^d$ as a subspace, $\iota(\mathbb{R}^d) \subset H$; the non-commutative multiplication structure, $\mathbf{u} \cdot \mathbf{v} \in H$ for $\mathbf{u}, \mathbf{v} \in H$; and a Hilbert space structure, $\langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{R}$ for $\mathbf{u}, \mathbf{v} \in H$.

Linear Functionals. Any $\boldsymbol{\ell} \in H$ can be treated as a linear functional by taking $\langle \boldsymbol{\ell}, \cdot \rangle : H \to \mathbb{R}$, we will primarily consider *finite linear functionals*

$$\boldsymbol{\ell} = (\boldsymbol{\ell}_0, \boldsymbol{\ell}_1, \dots, \boldsymbol{\ell}_M, 0, \dots) \in H,$$

which have only finitely many nonzero coordinates. Such finite linear functionals applied to our feature map $\widetilde{\varphi}(\mathbf{x}) \in H$ for sequences $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$, are rich enough so that

$$\mathbf{x} \mapsto \langle \boldsymbol{\ell}, \widetilde{\varphi}(\mathbf{x}) \rangle$$

can approximate any functions $f(\mathbf{x})$ of sequences $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$, and their collection characterizes the distribution of random sequences (i.e. random walks),

$$\mu \mapsto \{\langle \boldsymbol{\ell}, \mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})] \rangle : \boldsymbol{\ell} = (\boldsymbol{\ell}_0, \dots, \boldsymbol{\ell}_M, 0, \dots) \in H, M \geq 1\}$$

is injective; see Theorem 4.

C The Sequence Feature Map

In Section 2 we introduce the sequence feature map

$$\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H, \quad \widetilde{\varphi}(\mathbf{x}) = \varphi(\delta_0 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}),$$

for sequences in $\mathbb{R}^d$ where

$$\varphi : \mathbb{R}^d \to H$$

is an injective map from $\mathbb{R}^d$ into the algebra H. Here, $\delta_0 \mathbf{x} \coloneqq x_0$ and $\delta_i \mathbf{x} \coloneqq x_i - x_{i-1}$ for $i \geq 1$ denote the increments of a sequence $\mathbf{x} = (x_0, \ldots, x_k)$. Our main example is the tensor exponential

$$\varphi(x) = \exp_{\otimes}(x) = \left(\frac{x^{\otimes m}}{m!} \right)_{m \geq 0}.$$

The algebra H is the free algebra discussed in detail in Appendix B. In this case, a direct calculation shows that

$$\widetilde{\varphi}(\mathbf{x}) = \Big(\sum c(i_1, \ldots, i_m) \delta_{i_1} \mathbf{x} \otimes \cdots \otimes \delta_{i_m} \mathbf{x} \Big)_{m \geq 0}$$

where the sum is taken over $i_1 \leq \cdots \leq i_m$ with $i_1, \ldots, i_m \in \{0, \ldots, k\}$ and the coefficients $c(i_1, \ldots, i_m) \in \mathbb{R}$ can be computed in explicit form, see [62].

Universality and Characteristicness Universality and characteristicness follow for our "discrete time/sequence" signatures $\widetilde{\varphi}(\mathbf{x}) \in H$ from elementary arguments that we discuss below. In our setting it is convenient to include the sequence coordinate in the lifting, that is we consider sequences

$$\bar{\mathbf{x}} = (\bar{x}_0, \ldots, \bar{x}_k), \text{ where } \bar{x}_i = (i, x_i) \in \mathbb{R}^{d+1}. \tag{17}$$

Theorem 4. *The* time-parametrized sequence feature map

$$\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H, \quad \widetilde{\varphi}(\bar{\mathbf{x}}) = \varphi(\delta_0 \bar{\mathbf{x}}) \cdots \varphi(\delta_k \bar{\mathbf{x}}) \tag{18}$$

has the following properties:

1. ***Universality**: for any continuous*[7] $f : \mathsf{Seq}(\mathbb{R}^d) \to \mathbb{R}$*, any* $\epsilon > 0$*, and any compact set of sequences* $K \subset \mathsf{Seq}(\mathbb{R}^d)$*, there exists a* $\boldsymbol{\ell} = (\boldsymbol{\ell}_0, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m, 0, 0, \ldots)$ *such that*

$$\sup_{\mathbf{x} \in K} |f(\bar{\mathbf{x}}) - \langle \ell, \widetilde{\varphi}(\bar{\mathbf{x}}) \rangle| < \epsilon.$$

2. ***Characteristicness**: let* $\mathsf{Prob}(K)$ *be the set of probability measures that are supported on a compact subset* $K \subset \mathsf{Seq}(\mathbb{R}^d)$*. Then the map*

$$\mathsf{Prob}(K) \to H, \quad \mu \mapsto \mathbb{E}_{\mathbf{x} \sim \mu}[\widetilde{\varphi}(\bar{\mathbf{x}})]$$

is injective.

Proof. This is a folk theorem in control and probability theory: to see universality, it is sufficient to verify that $\{\langle \boldsymbol{\ell}, \widetilde{\varphi}(\bar{\mathbf{x}}) \rangle : \boldsymbol{\ell} = (\boldsymbol{\ell}_0, \ldots, \boldsymbol{\ell}_m, 0, \ldots), m \geq 0\}$ is a point-separating algebra for $C(K, \mathbb{R})$ since then the result follows from the Stone-Weierstrass theorem. Point separation follows from [22, Corollary 4.9], and a direct calculation shows the product of $\langle \boldsymbol{\ell}, \widetilde{\varphi}(\bar{\mathbf{x}}) \rangle$ and $\langle \boldsymbol{\ell}', \widetilde{\varphi}(\bar{\mathbf{x}}) \rangle$ is again a linear functional of $\widetilde{\varphi}(\bar{\mathbf{x}})$. This shows that the collection of functionals forms an algebra.

The characteristicness follows since $\mathsf{Prob}(K)$ is contained in the dual space of the space of continuous functions of sequences $C(K, \mathbb{R})$ and by universality, linear functionals of $\widetilde{\varphi}(\bar{\mathbf{x}})$ are dense in this space, so the result follows. $\square$

This result shows that:

- non-linear functions of sequences can be approximated using linear functionals in H; and
- the distribution of random sequences is characterized as the mean of our feature map.

In the terminology of statistical learning this says that $\widetilde{\varphi}$ is a universal and characteristic feature map for the set $\mathsf{Seq}(V)$ of sequences. For more background and extensions to non-compact sets of sequences or paths, we refer to [34, 16] and [6, Section 3.2]; for a more geometric picture see [39].

---

[7] We use the metric $d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{k} \|(x_i - y_i) - (x_{i-1} - y_{i-1})\|$ (if two sequences are of different lengths, we pad the end of the shorter one with the end point) to define the topology on $\mathsf{Seq}(\mathbb{R}^d)$.

Path Signatures. We now explain the remark made at the end of Section 2, that for the choice of φ as the tensor exponential (5), the resulting sequence feature map $\widetilde{\varphi}$ can be identified as the time discretization of a classical object in analysis, called the path signature. First, we lift a sequence $\mathbf{x} = (x_0, \dots, x_k)$ from discrete time to continuous time by identifying it as the piecewise linear path $\mathbf{X} = (\mathbf{X}(t))_{t \geq 0}$,

$$\mathbf{X}(t) \coloneqq x_i + (t - i)(x_{i+1} - x_i) \text{ for } t \in [i, i+1) \, .$$

Then a direct calculation shows that

$$\widetilde{\varphi}(\mathbf{x}) = \left(\int_0^k d\mathbf{X}^{\otimes m} \right)_{m \geq 0} \text{ where } \int_0^i d\mathbf{X}^{\otimes m} \coloneqq \int_{0 \leq t_1 \leq \dots \leq t_m \leq i} \dot{\mathbf{X}}(t_1) \otimes \dots \otimes \dot{\mathbf{X}}(t_m) dt_1 \cdots dt_m$$

and $i \in [0, k]$; note that $\dot{\mathbf{X}}(t)$ is well-defined for almost every $t \in [0, k]$ since $\mathbf{X}$ is piecewise linear, which is sufficient to make sense of this integral as a Riemann-Lebesgue integral (and stochastic integrals allow to treat rougher paths). Such sequences of iterated integrals are classical in analysis, probability theory, and control theory, and are known under various names (Path-ordered Exponential, Volterra series, Chen–Fliess Series, Chronological Exponential, etc.); we refer to them as *path signatures* as they are known in probability theory.

Diffusions and their Generator. The path signature can even can be defined for highly irregular paths such as Brownian motion by using stochastic (Ito-Stratonovich) integrals. This is the connection to our main motivation: if $\mathbf{X} = (\mathbf{X}_t)_{t \geq 0}$ is a Brownian motion in $\mathbb{R}^d$, then its generator is the classical Laplacian and the diffusion of Brownian particles is captured with the classical diffusion PDE, the heat equation. Gaveau [23] showed that if we lift Brownian trajectories $t \mapsto \mathbf{X}(t)$ evolving in the state space $\mathbb{R}^d$ into paths evolving in a richer state space H via the above signature construction,

$$t \mapsto \left(\int_0^t d\mathbf{X}^{\otimes m} \right)_{m \geq 0}, \tag{19}$$

then the stochastic process (19) is again a Markov process[8] and its generator satisfies a property called hypo-ellipticity (see the paragraph below). Our approach can be seen as the analogous construction on a graph and in discrete time: in the same way that (19) lifts Brownian motion from $\mathbb{R}^d$ to a process evolving in the state space H, the map $\widetilde{\varphi}$ lifts a simple random walk $(B_k)_{k \geq 0}$ on the graph to a random walk $\mathbf{L} = (L_k)_{k \geq 0}$ in H.

Hypo-elliptic Operators, Sub-elliptic Operators, and their Geometry. A full discussion is beyond the scope of this article and several books [31] have been written about this topic. For the interested reader, we give a very short informal picture on the space $\mathbb{R}^d$ which motivates our nomenclature but otherwise this paragraph can be safely skipped. A differential operator $\mathcal{L} = \sum \sigma_{i,i} \frac{\partial^2}{\partial x_i \partial x_j} + \sum_i \mu_i \frac{\partial}{\partial x_j}$ is called elliptic if the matrix $(\sigma_{i,j}(x))_{i,j}$ is invertible for all $x \in \mathbb{R}^d$. Every (time-homogenous) Markov process $\mathbf{X} = (\mathbf{X}_t)_{t \geq 0}$ gives rise to a differential operator

$$\mathcal{L}f(x) = \lim_{t \to 0} \frac{\mathbb{E}[f(\mathbf{X}_t) | \mathbf{X}_0 = x] - f(x)}{t}$$

but generically $\mathcal{L}$ is not elliptic. An important example class of Markov processes are stochastic differential equations,

$$d\mathbf{X}_t = \sum_{i=1}^{e} V_i(\mathbf{X}_t) d\mathbf{B}_t^i, \mathbf{X}_0 \in \mathbb{R}^d, \tag{20}$$

where $\mathbf{B}_t = (\mathbf{B}_t^1, \dots, \mathbf{B}_t^e)$ denotes a Brownian motion in $\mathbb{R}^e$ and $V_1, \dots, V_e$ are vector fields on $\mathbb{R}^d$. By identifying the vectors fields $V_1, \dots, V_e$ as differential operators, the generator of (20) can be written as

$$\mathcal{L} = \sum_{i=1}^{e} V_i^2, \tag{21}$$

---

[8] The process (19) is often called the canonical lift of Brownian motion to the free Lie group with d generators. Strictly speaking, Gaveau uses the first M iterated integrals, otherwise one deals with an "infinite-dimensional Lie group" which poses some technical challenges. This Lie group embeds into H and in this article we only use the structure of H and not the Lie group structure; see [23].

which is in general not elliptic. Sub- and hypo-ellipticity are properties of $\mathcal{L}$ that are weaker than ellipticty: ellipticity implies sub-ellipticity and, by a classic theorem of Hörmander, sub-ellipticity implies hypo-ellipticity. Hypo-ellipticy is in turn general enough to cover many important examples for which (sub-)ellipticity fails. For example, another celebrated result of Hörmander is that the SDE generator (21) is hypo-elliptic, whenever the Lie algebra generated by the vector fields and their brackets, spans at every point the full space $\mathbb{R}^d$. This not only allows us to study the properties of a large class of diffusions but also provides a natural link with geometry: the set of vector fields $V_1, \dots, V_e$ determines a subset of $\mathbb{R}^d$ that the SDE evolves on. For general hypo-elliptic operators, this set is not all of $\mathbb{R}^d$ and it is not even a Riemannian manifold; rather, it is a sub-Riemannian manifold. Intuitively, sub-Riemannian geometry is "rougher" than Riemannian geometry (e.g. no canonical connections exist) but still regular enough so that one can use geometric tools to study the underlying stochastic process and vice versa; see for example [59, 23]. This includes SDEs such as those evolving in free objects: for a natural choice of vector fields $V_1, \dots, V_e$ the solution of the SDE (20) is (19), see [23].

D Further Details on Hypo-elliptic Graph Diffusion

In Section 3, we introduced tensor analogues of classical matrix operators, which were then used to define hypo-elliptic graph diffusion. These tensor-valued matrices allow us to efficiently represent random walk histories on a graph, which can be manipulated using matrix operations. In this section, we discuss further details on the tensor adjacency matrix, provide a proof of Theorem 1, and introduce a variation of hypo-elliptic graph diffusion.

As in Section 3, we fix a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$, where the nodes are denoted by integers, $\mathcal{V} = \{1, \dots, n\}$, and $f : \mathcal{V} \to \mathbb{R}^d$ denotes the continuous node attributes.

Powers of the Tensor Adjacency Matrix. Recall that the powers of the classical adjacency matrix A counts the number of walks between two nodes on the graph. In particular, given $k \in \mathbb{N}$, and two nodes $i, j \in \mathcal{V}$, the result follows as a consequence of the sparsity pattern of A,

$$(A^k)_{i,j} = \sum_{i_1, \dots, i_{k-1}=1}^{n} A_{i,i_1} \cdot A_{i_1,i_2} \cdots A_{i_{k-1},j} = \sum_{i=i_0 \sim \dots \sim i_k = j} 1.$$

Note that the product $A_{i,i_1} \cdots A_{i_{k-1},j} = 0$ unless each pair of consecutive indices are adjacent in the graph, namely $i_{q-1} \sim i_q$ for all $q = 1, \dots, k$. Applying the same procedure to the tensor adjacency matrix from Equation (10), we obtain a summary of all walks between two nodes, rather than just the number of walks. In particular,

$$\begin{aligned}
(\widetilde{A}^k)_{i,j} &= \sum_{i=i_0 \sim \cdots \sim i_k = j} \widetilde{A}_{i,i_1} \cdot \widetilde{A}_{i_1,i_2} \cdots \widetilde{A}_{i_{k-1},j} \\
&= \sum_{i=i_0 \sim \cdots \sim i_k = j} \varphi(f(i_1) - f(i)) \cdot \varphi(f(i_2) - f(i_1)) \cdots \varphi(f(j) - f(i_{k-1})) \\
&= \sum_{i=i_0 \sim \cdots \sim i_k = j} \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}),
\end{aligned}$$

where $\mathbf{x} = (f(i_0), \dots, f(i_k))$ denotes the lifted sequence in the vector space $\mathbb{R}^d$. Note that this corresponds to the sequence feature map *without* the initial point $\delta_0 \mathbf{x}$.

Powers of the Tensor Transition Matrix. We now consider powers of the classical transition matrix $P = I - \mathcal{L}$, where $\mathcal{L} = I - D^{-1}A$ is the normalized graph Laplacian. The entries of P^k provide length k random walk probabilities; in particular, we have

$$(P^k)_{i,j} = \sum_{i=i_0 \sim \dots \sim i_k = j} \frac{1}{\deg(i_0)\deg(i_1)\dots\deg(i_{k-1})} = \mathbb{P}[B_k = j | B_0 = i].$$

The powers of the tensor transition matrix $\widetilde{P} = I - \widetilde{\mathcal{L}}$, where $\widetilde{\mathcal{L}} = I - D^{-1}\widetilde{A}$ is the hypo-elliptic Laplacian, will be the conditional expectation of the sequence feature map of the random walk process. In particular,

$$(\widetilde{P})^k_{i,j} = \sum_{i=i_0\sim\cdots\sim i_k=j} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x})\mathbb{P}[B_1 = i_1, B_2 = i_2, \ldots, B_k = i_k | B_0 = i]. \tag{22}$$

Proof of Hypo-elliptic Diffusion Theorem. Recall from Section 3 that the *hypo-elliptic graph diffusion equation* is defined by

$$\mathbf{v}_k - \mathbf{v}_{k-1} = -\widetilde{\mathcal{L}}\mathbf{v}_{k-1}, \quad \mathbf{v}_0 = \mathbb{1}_H,$$

where $\mathbb{1}_H^T = (1_H, \ldots, 1_H) \in H^n$ is the all-ones vector in H. Using the above computations for powers of the tensor transition matrix, we can prove Theorem 1, which is restated here.

Theorem 1. *Let* $k \in \mathbb{N}$, $\mathbf{L}_k = (L_0, \ldots, L_k)$ *be the lifted random walk from* (6)*, and* $\widetilde{P} = I - \widetilde{\mathcal{L}}$ *be the* tensor adjacency matrix. *The solution to the hypo-elliptic graph diffusion equation* (12) *is*

$$\mathbf{v}_k = (\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_0 = i])_{i=1}^n = \widetilde{P}^k\mathbb{1}_H.$$

Furthermore, if $F \in H^{n\times n}$ *is the diagonal matrix with* $F_{i,i} = \varphi(f(i))$*, then*

$$F\mathbf{v}_k = (\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)|B_0 = i])_{i=1}^n.$$

Proof of Theorem 1. We begin by proving the first equation. From the definition of hypo-elliptic diffusion, it is straightforward to see that

$$\mathbf{v}_k = (I - \widetilde{\mathcal{L}})\mathbf{v}_{k-1} = \widetilde{P}\mathbf{v}_{k-1} = \widetilde{P}^k\mathbf{v}_0 = \widetilde{P}^k\mathbb{1}_H.$$

We prove coordinate-wise that $\mathbf{v}_k = (\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_0 = i])_{i=1}^n$. Indeed, using the above and Equation (22), the i-th coordinate of $\mathbf{v}_k$ is

$$\begin{aligned}\mathbf{v}_k^{(i)} = (\widetilde{P}^k\mathbb{1}_H)^{(i)} = \sum_{j=1}^n(\widetilde{P}^k)_{i,j} &= \sum_{i=i_0\sim\ldots\sim i_k} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x})\mathbb{P}[B_1 = i_1, \ldots, B_k = i_k|B_0 = i] \\ &= \mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_0 = i],\end{aligned}$$

where $\mathbf{x} = (f(i_0), \ldots, f(i_k))$ is the lifted sequence corresponding to a walk $i_0 \sim \ldots \sim i_k$ on the graph. Next, we will also prove the second equation coordinate-wise. Using the above result, we have

$$(F\mathbf{v}_k)^{(i)} = \varphi(f(i))\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_0 = i] = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)|B_0 = i],$$

where we use the fact that $\delta_0\mathbf{L}_k = L_0 = f(i)$ when we condition $B_0 = i$. ☐

Forward Hypo-elliptic Diffusion. In the classical setting, we can consider both the forward and backward Kolmogorov equations. Throughout the main text and in the appendix so far, we have been considering the *backward* variants. In this section, we formulate the *forward* analogue of hypo-elliptic diffusion. In the classical graph setting, this corresponds to the following forward equation for $U_k \in \mathbb{R}^{n\times d}$ given by

$$U_k^T - U_{k-1}^T = -U_{k-1}^T\mathcal{L}, \quad U_0^{(i)} = f(i)$$

where the initial condition $U_0 \in \mathbb{R}^{n\times d}$ is specified by the node attributes. Note that because $P = D^{-1}A$ is right stochastic[9], this variation of the graph diffusion equation conserves mass in each coordinate of the node attributes at every time step.

Similarly we can formulate the forward hypo-elliptic graph diffusion equation for $\mathbf{v}_k \in H^n$ as

$$\mathbf{v}_k^T - \mathbf{v}_{k-1}^T = -\mathbf{v}_{k-1}^T\widetilde{\mathcal{L}}, \quad \mathbf{v}_0 = n^{-1}\mathbb{1}_H. \tag{23}$$

The solution of the Equation 23 is given below.

---

[9] Each column sums to one and hence multiplying on the left by a row vector conserves its mass.

Theorem 5. *Let $k \in \mathbb{N}$, $\mathbf{L}_k = (L_0, \ldots, L_k)$ be the lifted random walk from* (6)*, and $\widetilde{P} = I - \widetilde{\mathcal{L}}$ be the* tensor adjacency matrix. *The solution to the forward hypo-elliptic graph diffusion equation* (23) *is*

$$\mathbf{v}_k^T = (\mathbb{P}[B_k = i]\mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_k = i])_{i=1}^n = \frac{1}{n} \mathbb{1}_H^T \widetilde{P}^k.$$

Furthermore, if $F \in H^{n \times n}$ is the diagonal matrix with $F_{i,i} = \varphi(f(i))$, then

$$\frac{1}{n} \mathbb{1}_H^T F \widetilde{P}^k = (\mathbb{P}[B_k = i]\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k) | B_k = i])_{i=1}^n .$$

Proof. The proof proceeds in the same way as the backward equation. By definition of the forward hypo-elliptic diffusion, we have

$$\mathbf{v}_k^T = \mathbf{v}_0^T \widetilde{P}^k = \frac{1}{n} \mathbb{1}_H^T \widetilde{P}^k.$$

Now, recall that the initial point of the random walk process is chosen uniformly over all nodes; in other words, $\mathbb{P}[B_0 = i] = \frac{1}{n}$. Then, we show $\mathbf{v}_k^T = (\mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_k = i])_{i=1}^n$ coordinate-wise as

$$\begin{aligned}
\mathbf{v}_k^{(i)} &= \frac{1}{n} \sum_{j=1}^n (\widetilde{P}^k)_{j,i} \\
&= \sum_{i_0 \sim \ldots \sim i_k = i} \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}) \mathbb{P}[B_1 = i_1, \ldots, B_k = i | B_0 = i_0] \mathbb{P}[B_0 = i_0] \\
&= \sum_{i_0 \sim \ldots \sim i_k = i} \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}) \mathbb{P}[B_0 = j, \ldots, B_{k-1} = i_{k-1} | B_k = i] \mathbb{P}[B_k = i] \\
&= \mathbb{P}[B_k = i]\mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_k = i],
\end{aligned}$$

where $\mathbf{x} = (f(i_0), \ldots, f(i_k))$ is the lifted sequence corresponding to a walk $i_0 \sim \ldots \sim i_k$ on the graph. We will now prove the second equation. Note that we have

$$(F\widetilde{P}^k)_{i,j} = \sum_{i = i_0 \sim \ldots \sim i_k = j} \widetilde{\varphi}(\mathbf{x}) \mathbb{P}[B_1 = i_1, \ldots, B_k = i_k \,|\, B_0 = i].$$

Then, following the same reasoning as the first equation, we obtain the desired result. □

Weighted Graphs. In the prior discussion, we considered simple random walks in which the walk chooses one of the nodes adjacent to the current node uniformly at random. We can instead consider more general random walks on graphs, which can be described using a weighted adjacency matrix,

$$A_{i,j} = \begin{cases} c_{i,j} & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

In this case, we define the diagonal weighted degree matrix to be $D_{i,i} = \sum_{i \sim j} A_{i,j}$. We now define a weighted random walk $(B_k)_{k \geq 0}$ on the vertices $\mathcal{V}$ where the transition matrix is given by

$$P_{i,j} := D^{-1}A = \mathbb{P}(B_k = j \,|\, B_{k-1} = i) = \begin{cases} \frac{c_{i,j}}{\sum_{i \sim j'} c_{i,j'}} & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

With these weighted graphs, the powers of the adjacency and transition matrix can be interpreted in a similar manner as the standard case. Powers of the adjacency matrix provide total weights over walks, while the powers of the transition matrix provide the probability of a weighted walk going between two nodes after a specified number of steps. In particular,

$$\begin{aligned}
(A^k)_{i,j} &= \sum_{i \sim i_1 \sim \ldots \sim i_{k-1} \sim j} c_{i,i_1} \ldots c_{i_{k-1},j} \\
(P^k)_{i,j} &= \mathbb{P}[B_k = j \,|\, B_0 = i].
\end{aligned}$$

The tensor adjacency and tensor transition matrices are defined in the same manner as

$$\widetilde{A}_{i,j} := \begin{cases} c_{i,j}\varphi(f(j) - f(i)) & : i \sim j \\ 0 & : \text{otherwise} \end{cases}$$

$$\widetilde{P}_{i,j} := D^{-1}\widetilde{A} = \begin{cases} P_{i,j}\varphi(f(j) - f(i)) & : i \sim j \\ 0 & : \text{otherwise} \end{cases}$$

Note that powers of this weighted tensor transition matrix are exactly the same as the unweighted case from Equation (22), and thus the weighted version of both Theorem 2 and Theorem 5 immediately follow.

E Characterizing Random Walks

In this appendix, we will provide further details on how the features obtained via hypo-elliptic diffusion are able to characterize the underlying random walk processes. These results rely on the characteristic property of the time-parametrized sequence feature map from Theorem 4.

Computation with Time Parametrization. Recall that the time-parametrized sequence feature map from Equation (18) applies the algebra lifting to the time-parametrized sequence $\bar{\mathbf{x}} := (\bar{x}_0, \dots, \bar{x}_k)$, where $\bar{x}_i = (i, x_i)$. Note that we have $\delta_i \bar{\mathbf{x}} = (1, x_i - x_{i-1})$. Thus, the hypo-elliptic diffusion equations and the low-rank algorithm given in Theorem 3 easily extends to this setting.

Characterizing Random Walks. Recall that hypo-elliptic diffusion yields a feature map for labelled graphs by mean-pooling the individual node features as

$$\Psi(\mathcal{G}) = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)],$$

where $\mathbf{L}_k = (L_0, \dots, L_k)$ is the lifted random walk process in $\mathbb{R}^d$. We will now prove Theorem 2, which we restate here with more details.

Theorem 2. *Let*

$$\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H, \quad \widetilde{\varphi}(\bar{\mathbf{x}}) = \varphi(\delta_0 \bar{\mathbf{x}}) \cdots \varphi(\delta_k \bar{\mathbf{x}}),$$

where $\bar{\mathbf{x}}$ appends the time parametrization to the sequence $\mathbf{x}$ as in Equation (17). Furthermore, suppose we have the resulting graph feature map

$$\Psi(\mathcal{G}) = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)].$$

Let $\mathcal{G}$ and $\mathcal{G}'$ be two labelled graphs, and $\mathbf{L}_k = (L_0, \dots, L_k)$ and $\mathbf{L}'_k = (L'_0, \dots, L'_k)$ be the k-step lifted random walk as defined in Equation (6), given the random walk processes B and B' on $\mathcal{G}$ and $\mathcal{G}'$ respectively. Then, $\Psi(\mathcal{G}) = \Psi(\mathcal{G}')$ if and only if the distributions of $\mathbf{L}_k$ and $\mathbf{L}'_k$ are equal.

Proof. First, if the distributions of the two random walks $\mathbf{L}_k$ and $\mathbf{L}'_k$ are equal, then it is clear that $\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)] = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}'_k)]$.

Next, suppose $\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)] = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}'_k)]$. Then, note that the random walk distributions are finitely supported (hence compactly supported) distributions, and thus by Theorem 4, they must be equal. ☐

This result shows that the hypo-elliptic diffusion completely characterizes random walk histories; thus providing a highly descriptive summary of labelled graphs. There is an analogous result for the individual node features in terms of conditional expectations.

Theorem 6. *Let*

$$\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H, \quad \widetilde{\varphi}(\bar{\mathbf{x}}) = \varphi(\delta_0 \bar{\mathbf{x}}) \cdots \varphi(\delta_k \bar{\mathbf{x}}),$$

where $\bar{\mathbf{x}}$ appends the time parametrization to the sequence $\mathbf{x}$ as in Equation (17). Furthermore, suppose we have the resulting node feature map

$$\Phi(i) = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k) | B_0 = i].$$

Let $\mathcal{G}$ and $\mathcal{G}'$ be two labelled graphs, and $\mathbf{L}_k = (L_0, \dots, L_k)$ and $\mathbf{L}'_k = (L'_0, \dots, L'_k)$ be the k-step lifted random walk as defined in Equation (6), given the random walk processes B and B' on $\mathcal{G}$ and $\mathcal{G}'$ respectively. Then for two nodes $i \in \mathcal{V}$ and $i' \in \mathcal{V}'$ on the two respective graphs, $\Phi(i) = \Phi(i')$ if and only if the conditional distributions of $\mathbb{P}[\mathbf{L}_k \mid B_0 = i]$ and $\mathbb{P}[\mathbf{L}'_k \mid B'_0 = i']$ are equal.

Proof. The proof is analogous to the proof of Theorem 2 given above. ☐

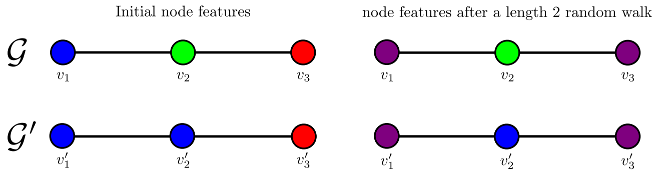

Figure 1: On the left: the graphs $\mathcal{G}$ and $\mathcal{G}'$ with their initial labels. On the right: the labels of $\mathcal{G}$ and $\mathcal{G}'$ after random walk of length 2.

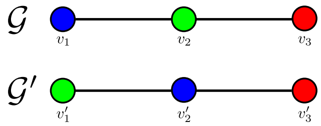

Figure 2: Here the global graph features computed using classical diffusion cannot distinguish between the graphs $\mathcal{G}$ and $\mathcal{G}'$, however using our framework creates global graph features that can distinguish between the two.

Examples. As mentioned after Theorem 2 in the main text, the benefits of capturing path-dependence can be significant. In addition to the experimental results one can gain intuition for this by looking at simple toy examples. For example, consider as labels RGB colors, that is $f : \mathcal{V} \to [0,1]^3 \subset \mathbb{R}^3$ and random walks of length $k = 2$ on the two graphs $\mathcal{G}, \mathcal{G}'$ with vertex sets $\mathcal{V} = \{v_1, v_2, v_3\}$, resp. $\mathcal{V}' = \{v_1', v_2', v_3'\}$.

We will now see two examples where features computed using classical diffusion cannot distinguish either the nodes or the graphs, but features computed using hypo-elliptic diffusion can.

Consider colors $a, b, c \in \mathbb{R}^3$ as labels on the nodes of a graph with three nodes and two edges as are represented in Figures 1 and 2, with colors a and c on the end nodes and b on the middle node. After a random walk B_2 of length two, simple calculations show that the labels on the nodes will be $\frac{1}{2}(a + c)$ on the two end nodes, and b on the middle node.

The first example in Figure 1 considers node features. If one only has access to the marginal distribution $L_2 = f(B_2)$ to construct node features, then the left-most and right-most nodes in the graphs $\mathcal{G}$ and $\mathcal{G}'$ from Figure 1 are indistinguishable, that is

$$\mathbb{E}[L_2|B_0 = v_1] = \mathbb{E}[L_2'|B_0' = v_1'] = \frac{1}{2}(r + b).$$

In stark contrast, the laws of (L_0, L_1, L_2) and (L_0', L_1', L_2') clearly differ since L_1, resp. L_1', are with probability one equal to green, resp. blue. Thus Theorem 2 guarantees that

$$\Phi(v_1) = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)|B_0 = v_1] \neq \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k')|B_0' = v_1'] = \Phi(v_1').$$

In fact, for this example, a direct calculation shows that

$$\langle e_b \otimes e_g, \widetilde{\varphi}(\mathbf{L}_2) \rangle = 1 \neq 0 = \langle e_b \otimes e_g, \widetilde{\varphi}(\mathbf{L}_2') \rangle.$$

The second example in Figure 2 considers graph features. The global graph features for 2-step classical diffusion is computed as the average of the node features, i.e. the global graph feature is $\frac{1}{3}(a + b + c)$. Hence, the graphs of Figure 2 are not distinguishable through features computed from standard diffusion. However, using hypo-elliptic diffusion allows us to take into account the order in which the colors appear in the random walk. Indeed the colors red and green never appear consecutively in the graph $\mathcal{G}'$, hence the $e_r \otimes e_g$ component of $\Psi(\mathcal{G}')$ is zero; in other words,

$$\langle e_r \otimes e_g, \Psi(\mathcal{G}) \rangle = 1 \neq 0 = \langle e_r \otimes e_g, \Psi(\mathcal{G}') \rangle,$$

showing that $\Psi(\mathcal{G}) \neq \Psi(\mathcal{G}')$.

Simple examples as the above demonstrate how using only the marginal distribution after k steps lacks expressive power for both node and graph features. Therefore, while classical diffusion smooths out neighbourhood features, hypo-elliptic diffusion retains finer descriptions of local neighbourhoods.

F Details on the Low Rank Algorithm.

In this section, we will provide further details and proofs on the low-rank approximation method discussed in Section 4. We can use low-rank tensors to define corresponding low-rank functionals of the features obtained via hypo-elliptic diffusion. The following is the definition of *CP-rank* of a tensor from [10].

Definition 1. The *rank* of a level m tensor $\mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}$ is the smallest number $r \geq 0$ such that we can express $\mathbf{v}_m$ as

$$\mathbf{v}_m = \sum_{i=1}^{r} v_{i,1} \otimes \ldots \otimes v_{i,m}, \quad v_{i,j} \in \mathbb{R}^d.$$

We say that $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots) \in H$ is *rank* 1 if all $\mathbf{v}_m$ are rank 1 tensors.

We will now prove Theorem 3, which is stated using the node feature map without `ZeroStart` (see Appendix G). In particular, the node feature map is given by

$$\hat{\Phi}_k(i) = \mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) \,|\, B_0 = i] = (\widetilde{P}^k \mathbb{1}_H)^{(i)} \in H, \tag{24}$$

where we explicitly specify the walk length k in the subscript. Furthermore, if we also need to specify the tensor degree m, we will use two subscripts, where

$$\hat{\Phi}_{k,m}(i) \in (\mathbb{R}^d)^{\otimes m}$$

is the level m component of the hypo-elliptic diffusion with a walk length of k. Throughout the proof, we omit the H subscript for the all-ones vector, such that $\mathbb{1}^T := (1_H, \ldots, 1_H) \in H^n$, and we denote $\mathbb{1}_i^T = (0, \ldots, 1_H, \ldots, 0) \in H^n$ to be the unit vector in the i^{th} coordinate.

Proof of Theorem 3. First, we will show that $f_{1,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_1(i) \rangle$ for all $m = 1, \ldots, M$. By the definition of hypo-elliptic diffusion, we know that

$$\hat{\Phi}_1(i) = \mathbb{1}_i^T \widetilde{P} \mathbb{1} = \sum_{j=1} \widetilde{P}_{i,j} = \sum_{i \sim j} \frac{\exp_{\otimes}(f(j) - f(i))}{d_i}.$$

By explicitly expressing the level m component, and by factoring out the inner product, we get

$$\begin{aligned}\left\langle \boldsymbol{\ell}_m, \frac{\exp_{\otimes}(f(j) - f(i))}{d_i} \right\rangle &= \frac{1}{d_i m!} \prod_{r=M-m+1}^{M} \langle u_r, f(j) - f(i) \rangle \\ &= \frac{1}{m!} (P_{i,j} \cdot C_{i,j}^{u_{M-m+1}} \cdot \ldots \cdot C_{i,j}^{u_M}).\end{aligned}$$

Then by linearity of the inner product, we get $f_{1,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_1(i) \rangle$.

Next, we continue by induction and suppose that $f_{k-1,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_{k-1}(i) \rangle$ holds for all $m = 1, \ldots, M$. Starting once again from the definition of hypo-elliptic diffusion, we know that

$$\hat{\Phi}_k(i) = \mathbb{1}_i^T \widetilde{P} \hat{\Phi}_{k-1} = \sum_{i \sim j} \widetilde{P}_{i,j} \cdot \hat{\Phi}_{k-1}(j).$$

Fix a degree m. We explicitly write out the level m component of this equation by expanding $\widetilde{P}_{i,j}$ and the tensor product as

$$\begin{aligned}\hat{\Phi}_{k,m}(i) &= \sum_{i \sim j} \sum_{r=0}^{m} \frac{(f(j) - f(i))^{\otimes r}}{d_i r!} \cdot \hat{\Phi}_{k-1,m-r}(j) \\ &= \sum_{i \sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} + \sum_{r=1}^{m} \frac{1}{r!} \sum_{i \sim j} \frac{(f(j) - f(i))^{\otimes r}}{d_i} \cdot \hat{\Phi}_{k-1,m-r}(j)\end{aligned} \tag{25}$$

Note that the first sum is equivalent to

$$\sum_{i \sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} = \mathbb{1}_i^T P \cdot \hat{\Phi}_{k-1,m}.$$

Applying the linear functional $\boldsymbol{\ell}_m$ and the induction hypothesis to this, we have

$$\left\langle \boldsymbol{\ell}_m, \sum_{i \sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} \right\rangle = \mathbb{1}_i^T P \cdot f_{k-1,m}.$$

For the second sum in Equation (25), we can factor the inner product and apply the induction hypothesis to get

$$\left\langle \boldsymbol{\ell}_m, \sum_{i \sim j} \frac{(f(j) - f(i))^{\otimes r}}{d_i} \cdot \hat{\Phi}_{k-1,m-r}(j) \right\rangle = \sum_{j=1}^{n} P_{i,j} \cdot C_{i,j}^{u_{M-m+1}} \cdot \ldots \cdot C_{i,j}^{M-m+r} \cdot f_{k-1,m-r}.$$

Putting this all together, we get

$$\hat{\Phi}_{k,m}(i) = \mathbb{1}_i^T \left(P \cdot f_{k-1,m} + \sum_{r=1}^{m} \frac{1}{r!} (P \odot C^{u_{M-m+1}} \odot \ldots \odot C^{u_M}) \cdot f_{k-1,m-r} \right) = f_{k,m}(i).$$

□

Computing the ZeroStart Variation. We can adapt the recursive algorithm provided by Theorem 3 above in order to compute low rank approximations to the ZeroStart variation of the node features, which is used throughout the main text (see also Appendix G). In particular, we consider

$$\Phi_k(i) = \mathbb{E}[\varphi(\delta_0 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) \,|\, B_0 = i] = \varphi(\delta_0 \mathbf{L}_k) \cdot \hat{\Phi}_k(i), \tag{26}$$

where $\hat{\Phi}_k$ is the variation without ZeroStart defined in Equation (24). Note that we can factor the $\varphi(\delta_0 \mathbf{L}_k)$ term out of the expectation due to the conditioning $B_0 = i$, and thus $\varphi(\delta_0 \mathbf{L}_k) = \varphi(f(i))$ is fixed. Thus, we can compute low rank functionals of $\Phi_k(i)$ using one additional step.

Theorem 7. *Using the same hypotheses as Theorem 3, let*

$$\boldsymbol{\ell}_m = u_{M-m+1} \otimes \ldots \otimes u_M,$$

where $u_m \in \mathbb{R}^d$ *for* $m = 1, \ldots, M$ *and let*

$$F_i^u \coloneqq \langle u, f(i) \rangle.$$

Then,

$$\langle \boldsymbol{\ell}_M, \Phi_k(i) \rangle = \sum_{r=0}^{M} \frac{1}{m!} F_i^{u_1} \cdots F_i^{u_r} \cdot f_{k,M-r}(i),$$

where $f_{k,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_k(i) \rangle$ *from Theorem 3.*

Proof. Using the definition of $\Phi_k(i)$ from Equation (26), and expanding out the definition of $\varphi(\delta_0 \mathbf{L}_k)$ at level M, we have

$$\Phi_{k,M}(i) = \sum_{r=0}^{M} \frac{f(i)^{\otimes r}}{r!} \cdot \hat{\Phi}_{k,M-r}(i).$$

Then, taking the linear functional and distributing, we obtain the result

$$\langle \boldsymbol{\ell}_M, \Phi_k(i) \rangle = \sum_{r=0}^{M} \frac{1}{r!} F_i^{u_1} \cdots F_i^{u_r} \cdot f_{k,M-r}(i).$$

□

Algorithm 1 Computing a rank-1 functional of the hypoelliptic node features.

```
**Input:** Graph $\mathcal{G} = ([n], \mathcal{E}, f)$ with adjacency matrix $A$ and transition matrix $P$
a rank-1 tensor $\boldsymbol{\ell} = u_1 \otimes \cdots \otimes u_M \in T(\mathbb{R}^d)$, truncation level $M \in \mathbb{N}$, walk length $k \in \mathbb{N}$
Compute $C[m,i,j] \leftarrow A[i,j] \cdot \langle u_m, f(j) - f(i) \rangle$ for $m \in [M]$, $r \in [R]$, and $i, j \in [n]$
for $i = 1$ to $k$ do
    parfor $m = 1$ to $M$ do
        if $i == 1$ then
            Assign $Q[1,m,:] \leftarrow \frac{1}{m!}(P \odot C[M-m+1,:,:] \odot \ldots \odot C[M,:,:]) \cdot \mathbb{1}$
        else
            Assign $Q[i,m,:] \leftarrow P \cdot Q[i-1,m,:] + \sum_{j=1}^{m} \frac{1}{j!}(P \odot C[M-m+1,:,:] \odot \ldots \odot C[M-m+j]) \cdot Q[i-1,m-j,:]$
        end if
    end parfor
end for
**Output:** Vector $Q[k,m,:]$ of length $n$ representing node features $(\langle \boldsymbol{\ell}, \hat{\Phi}_k(i) \rangle)_{i=1,\ldots,n}$
```

Computational Complexity. We will now consider the computational complexity of our algorithms. We begin by noting that the naive approach of computing $\Phi_k(i) = (F\widetilde{P}^k \mathbb{1}_H)^{(i)}$ has the computational complexity of matrix multiplication; though this counts tensor operations, which itself requires $O(d^m)$ scalar multiplications at tensor degree m. This is computationally too expensive for practical applications.

Next, we consider the complexity of the recursive low-rank algorithm from Theorem 3, where the primary computational advantage is the fact that we only perform *scalar* operations rather than *tensor* operations. We consider the recursive step from Equation (15), reproduced here for $m = M$,

$$f_{k,M} \coloneqq P \cdot f_{k-1,M} + \sum_{r=1}^{M} \frac{1}{r!} \left(P \odot C^{u_1} \odot \cdots \odot C^{u_r}\right) \cdot f_{k-1,M-r}.$$

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ with $n = |\mathcal{V}|$ nodes and $E = |\mathcal{E}|$ edges, both P and C^u are sparse $n \times n$ (scalar) matrices with $O(E)$ nonzero entries, and $f_{k,m}$ is an n-dimensional column vector. Recall that $\odot$ denotes element-wise multiplication, and thus both the sparse matrix-matrix multiplication and the sparse matrix-vector multiplication have complexity $O(E)$. Furthermore, the entry-wise products $C^{u_1} \odot \cdots \odot C^{u_r}$ differ by only one factor between $r = m$ and $r = m+1$, and thus, computing $f_{k,M}$ assuming all lower $f_{k',m'}$ have been computed has complexity $O(ME)$. Taking into account the two recursive parameters results in a complexity of $O(kM^2E)$. Note that this is the complexity to compute features for *all nodes*.

Once the $f_{k,m}$ are computed, the complexity of adding the start point from Theorem 7 is $O(M)$.

Pseudocode. We provide pseudocode in Algorithm 1 to demonstrate the implementation of Theorem 3. The primary idea is to recursively compute the $f_{k,m}$ vector from Theorem 3 (renamed as an array $Q[:,:,:]$ to avoid confusion with the node features f). In particular, note that the inner loop over m can be parallelized. Thus, while the theoretical complexity in terms of max tensor level is $O(M^2)$ as discussed above, when we run the algorithm in parallel on a GPU, the computation time scales roughly linearly, as shown empirically in Table 7 in Appendix H.4.

G Variations and Hyperparameters of Hypo-Elliptic Diffusion

In this appendix, we summarize possible variations of the sequence feature map, leading to different hypo-elliptic diffusion features. The choice of variation is learned during training, and we also summarize the hyperparameters used for our features. While the theoretical results on characterizing random walks, such as Theorem 2, depend on specific choices of the sequence feature map, there exist analogous results for these variations, which can characterize random walks up to certain equivalences. Furthermore, the computation of these variations can be performed in the same way: through tensorized linear algebra for exact solutions, and through an analogous low-rank method (as in Theorem 3) for approximate solutions.

We fix an algebra lifting $\varphi : \mathbb{R}^d \to H$ and let $\mathbf{x} = (x_0, \ldots, x_k) \in \mathsf{Seq}(\mathbb{R}^d)$. The simplest sequence feature map to define simply multiplies the terms in the sequence together as

$$\widetilde{\varphi}(\mathbf{x}) = \varphi(x_0) \cdots \varphi(x_k). \tag{27}$$

Note that this is *not* the sequence feature map used in the main text. We will now discuss several variations of this map, where $\widetilde{\varphi}_{\text{inc,zs}}$ is the one primarily used in the main text and $\widetilde{\varphi}_{\text{inc,zs,tp}}$ is used to characterize random walks in Theorem 2 and Appendix E.

Increments (`Diff`). Rather than directly multiplying terms in the sequence together, we can instead multiply the *increments* as

$$\widetilde{\varphi}_{\text{inc}}(\mathbf{x}) = \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}),$$

where $\delta_i \mathbf{x} \coloneqq x_i - x_{i-1}$ for $i \geq 1$. In both cases, the sequence feature map is the path signature of a continuous piecewise-linear path when we set $\varphi = \exp_\otimes$, as discussed in Appendix C, and it is instructive to use this perspective to understand the effect of increments. If we use increments, the path corresponding to the sequence is

$$\mathbf{X}_{\text{inc}}(t) \coloneqq x_i + (t-i)(x_{i+1} - x_i) \text{ for } t \in [i, i+1)\,,$$

while if we do not use increments, the path corresponding to the sequence is

$$\mathbf{X}(t) \coloneqq \sum_{j=0}^{i-1} x_j + (t-i)x_i \text{ for } t \in [i, i+1)\,.$$

Thus, when we use increments the sequence $\mathbf{x}$ corresponds to the vertices of the path $\mathbf{X}_{\text{inc}}$, while if we do not, it corresponds to the vectors between vertices of the path $\mathbf{X}$. In practice, this variation corresponds to taking first-differences of the sequence $\mathbf{x}$ before using eq. (27).

Zero starting point (`ZeroStart`). The sequence feature map with increments, $\widetilde{\varphi}_{\text{inc}}$, as defined above is *translation-invariant*, meaning $\widetilde{\varphi}_{\text{inc}}(\mathbf{x} + a) = \widetilde{\varphi}_{\text{inc}}(\mathbf{x})$, where $\mathbf{x} + a = (x_0 + a, x_1 + a, \ldots, x_k + a)$ for some $a \in \mathbb{R}^d$. In order to remove translation invariance, we can start each sequence at the origin $0 \in \mathbb{R}^d$ by pre-appending a 0 to each sequence. A concise way to define the resulting *zero started* sequence feature map is

$$\widetilde{\varphi}_{\text{inc,zs}}(\mathbf{x}) = \varphi(\delta_0 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}),$$

where we define $\delta_0 \mathbf{x} \coloneqq x_0$. This is the sequence feature map defined in Equation (1). Note that this variation does not change the sequence feature map if we do not use increments.

Time parametrization. When we relate sequences to piecewise linear paths as described in Appendix C, we can use the fact that the path signature is invariant under reparametrization, or more generally, tree-like equivalence [28]. In terms of discrete sequences, this includes invariance with respect to 0 elements in the sequence (without increments), and repeated elements in the sequence (with increments). In order to remove this invariance, we can include *time parametrization* by setting

$$\widetilde{\varphi}_{-,\text{tp}}(\mathbf{x}) \coloneqq \widetilde{\varphi}_{-}(\bar{\mathbf{x}}),$$

where $\bar{\mathbf{x}} \coloneqq (\bar{x}_0, \ldots, \bar{x}_k) \in \mathsf{Seq}(\mathbb{R}^{d+1})$, with $\bar{x}_i \coloneqq (i, x_i) \in \mathbb{R}^{d+1}$. This is a simple form of positional encoding, but other encodings are also possible, e.g. sinusoidal waves as in [64].

Algebra lifting (`AlgOpt`). Throughout this article, we have used the tensor exponential as the algebra lifting. However, we can also scale each level of the lifting independently, and keep these as hyperparameters to optimize. In particular, for a sequence $\mathbf{c} = (c_0, c_1, \ldots) \in \mathbb{R}^{\mathbb{N}}$, we define $\varphi^{\mathbf{c}} : \mathbb{R}^d \to H$ to be

$$\varphi^{\mathbf{c}}(x) \coloneqq \left(c_m x^{\otimes m}\right)_{m=0}^{\infty},$$

where $1/m!$ is used as initialization for c_m and learned along with the other parameters.

The choice of which variant of the sequence feature map to use depends on which invariance properties are important for the specific problem. In practice, the choice can be learned during the training, which is done in our experiments in Section 5. Furthermore, the features obtained through the low-rank hypo-elliptic diffusion depend on three hyperparameters:

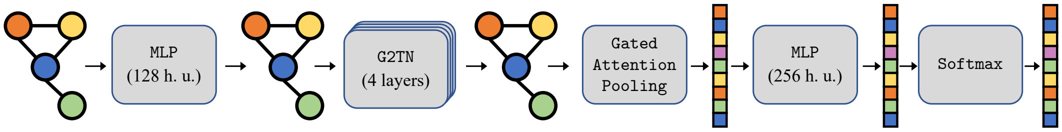

Figure 3: Visualization of the architecture used for NCI1 and NCI109 described in Section 5.

- the length of random walks;
- the number of low-rank functionals;
- the maximal tensor degree;
- the number of iterations (layers).

Note that the first three hyperparameters can also potentially vary across different iterations.

H Experiments

We have implemented the low-rank algorithm for our layers given in Theorem 3 using Tensorflow, Keras and Spektral [25]. Code is available at `https://github.com/tgcsaba/graph2tens`. All experiments were run on one of 3 computing clusters that were overall equipped with 5 NVIDIA Geforce 2080Ti, 2 Quadro GP100, 2 A100 GPUs. The largest GPU memory allocation any one of the experiments required was around $\sim$5GB. For the experiments ran using different random seeds, the seed was used to control the randomness in the (1) data splitting process, (2) parameter initialization, (3) optimization procedure. For an experiment with overall n_{runs} number of runs, the used seeds were $\{0, \ldots, n_{\text{runs}} - 1\}$.

H.1 Model details.

The architecture described in Section 5 is visualized conceptually on Figure 3, where for `G2T(A)N` the `G2TN` stack is replaced by the attentional variation without changing any hyperparameters. Further to the discussion given in the main text, here we provide more details on the model implementation.

Initialization. In our `G2TN` and `G2T(A)N` layers, we linearly project tensor-valued features using linear functionals that use the same low-rank parametrization as the *recursive* variation in [62, App. D.2]. There, the authors also propose in Appendix D.5 an initialization heuristic for setting the variances of these component vectors, that we have employed with a centered uniform distribution.

Regularization. We also apply ℓ_2 regularization for both experiments in Sections 5 and H.3 for which we discuss the implementation here. Given a max. tensor degree $M \geq 1$ and a max. tensor rank $R \geq 1$ representing the layer width, there are overall RM rank-1 linear functionals of the form

$$\boldsymbol{\ell}_m^r = u_{M-m+1}^r \otimes \cdots \otimes u_M^r \quad \text{for } m = 1, \ldots M \text{ and } r = 1, \ldots, R. \tag{28}$$

Since in practice these are represented using only the component vectors $u_i^r \in \mathbb{R}^d$, a naive application of ℓ_2 regularization would lead to computing the 2-norm of each component vector u_i^r to penalize the loss function. However, we found this approach to underperform compared to the following idea. Although our algorithm represents the functionals using a rank-1 decomposition and computes the projection of each tensor-valued node feature without explicitly building high-dimensional tensors, conceptually we still have tensors ($\boldsymbol{\ell}_m^r$) acting on tensors ($\Phi(i)$ for $i \in \mathcal{V}$), and hence the tensor norm, $\|\boldsymbol{\ell}_m^r\|_2$, should be used for regularization. Fortunately, this can also be computed efficiently:

$$\|\boldsymbol{\ell}_m^r\|_2 = \left\|u_{M-m+1}^r \otimes \cdots \otimes u_M^r\right\|_2 = \left\|u_{M-m+1}^r\right\|_2 \cdots \|u_M^r\|_2 .$$

Further, as is common, we replace the ℓ_2 norm with its squared value, sum over all functionals in (28), and multiply by the regularization parameter $\lambda > 0$ so that the final penalty is given by

$$\texttt{L2Penalty} = \lambda \sum_{r=1}^{R} \sum_{m=1}^{M} \left\|u_{M-m+1}^r\right\|_2^2 \cdots \|u_M^r\|_2^2 .$$

H.2 Experiment Details.

Here we provide further details on the main experiment described in Section 5.

Hyperparameter Selection. The model architecture was primarily motivated by GraphTrans (small) from [70]. Specifically, the number and width of GNN layers, and the dropout rate was adopted as is. The ℓ_2 regularization strength was chosen equal to the weight decay rate. The other hyperparameters were tuned on a single split of NCI1. For the random walk length, we experimented with values $2, 5, 10$. For the given GNN depth (4), 2 RW steps per layer was not enough to learn long-range interactions, while 10 significantly slowed down the convergence rate during training. For the maximal degree of tensors, we experimented with values from $2, 3, 4$, and using values beyond 2 did not provide improvements. Intuitively, the tensor degree represents the order of nonlinear interactions that are learnable by the layer, e.g. a degree of 2 encodes pairwise interactions between node features in the neighbourhood, while a higher degree of M allows to encode interactions between certain M-tuples of nodes. We suggest that a degree of 2 is a good baseline setting, and that increasing the GNN depth instead allows to efficiently capture higher order interactions, while increasing the effective influence radius at the same time. For the pre- and postprocessing layers, we experimented with various depths and choosing more than 1 layer for each was counterproductive. The number of units was simply set to the GNN width (128) in the preprocessing layer, while for the postprocessing layer slightly increasing it was found to provide improvements (256), potentially to compensate for the large amount of information that is compressed in the pooling step.

Table 2: Accuracies computed over 5 seeds of `G2T(A)N` ablated by changing a single option.

Dataset	`NoDiff`	`NoZeroStart`	`NoAlgOpt`	`NoJK`	`NoSkip`	`NoNorm`	`AvgPool`
NCI1	80.1 ± 0.7	79.5 ± 1.8	81.6 ± 1.6	82.1 ± 1.8	81.8 ± 0.9	81.6 ± 1.5	82.4 ± 0.9
NCI109	78.2 ± 1.2	77.7 ± 1.8	77.5 ± 1.3	77.6 ± 1.2	78.3 ± 1.3	79.8 ± 1.4	77.6 ± 1.3

Ablation Studies. Further to using attention, we give a list of ideas for variations on our models in Appendix G, which can be summarized briefly as: (i) using increments of node attributes (`Diff`), (ii) preprending a zero point to sequences (`ZeroStart`), (iii) optimizing over the algebra embedding (`AlgOpt`), all of which are built into our main models. Further, the previous architectural choices aimed at incorporating several commonly used tricks for training GNNs. We investigate the effect of the previous variations and ablate the architectural "tricks" by measuring the performance change resulting from ceteris paribus removing it. Table 2 shows the result for `G2T(A)N` . To summarize the main observations, the model is robust to all the architectural changes, removing the layer norm even improves on NCI109. Importantly, replacing the attention pooling with mean pooling does not significantly affect the performance, but actually slightly improves on NCI1. Regarding variations, `AlgOpt` slightly improves on both datasets, while removing `Diff` and `ZeroStart` significantly degrades the accuracy on NCI1. The latter means that translations of node attributes are important, not just their distances.

We give the analogous ablation study for the `G2TN` model in Table 3, and compare the derived conclusions between the attentional and attention-free versions. First, we discuss the layer variations. Similarly to `G2T(A)N` , `NoDiff` slightly decreases the accuracy. However the conclusions regarding `NoZeroStart` and `NoAlgOpt` are different. In this case, removing `ZeroStart` actually improves the performance, while on `G2T(A)N` the opposite was true. An interpretation of this phenomenon is that only the attention mapping that is used to learn the random walks required information about translations of the layer's features, and not the tensor-features themselves. Another difference is that `NoAlgOpt` degrades the accuracy more significantly for `G2TN` . A possible explanation is that since `G2T(A)N` layers are more flexible thanks to their use of attention, they rely less on being able to learn the algebraic lift, while as `G2TN` layers are more rigid in their random walk definition, and benefit more from the added flexibility of `AlgOpt`. Additionally, it seems that `G2TN` is more sensitive to the various architectural options, and removing any of them, i.e. `NoJK`, `NoSkip`, `NoNorm` or `AvgPool`, degrades the accuracy by 1% or more on at least one dataset. Intuitively, it seems that overall the `G2T(A)N` model is more robust to the various architectural "tricks", and more adaptable due to its ability to learn the the random walk.

Table 3: Accuracies computed over 5 seeds of `G2TN` ablated by changing a single option.

Dataset	`NoDiff`	`NoZeroStart`	`NoAlgOpt`	`NoJK`	`NoSkip`	`NoNorm`	`AvgPool`
NCI1	79.1 ± 1.5	80.7 ± 0.6	78.9 ± 1.3	79.4 ± 1.9	81.0 ± 1.3	79.5 ± 1.1	80.3 ± 1.7
NCI109	77.8 ± 1.2	79.5 ± 1.5	76.5 ± 1.7	78.1 ± 1.9	77.0 ± 1.6	77.4 ± 2.7	77.6 ± 1.8

Table 4: Accuracies of our models on the citation datasets computed over 100 seeds compared with the 4 consistently best performing baselines from [57].

Dataset	**GCN**	**GAT**	**MoNet**	**GraphSage (mean)**	**`G2TN` (ours)**	**`G2T(A)N` (ours)**
Cora	81.5 ± 1.3	81.8 ± 1.3	81.3 ± 1.3	79.2 ± 7.7	$\mathbf{82.6 \pm 1.0}$	82.0 ± 1.1
Citeseer	$\mathbf{71.9 \pm 1.9}$	71.1 ± 1.9	71.2 ± 2.0	71.6 ± 1.9	69.4 ± 1.0	68.2 ± 1.3
Pubmed	77.8 ± 2.9	78.7 ± 2.3	78.6 ± 2.3	77.4 ± 2.2	$\mathbf{78.8 \pm 1.9}$	78.0 ± 1.9

H.3 Further Experiments

Citation datasets. Additionally to transductive learning on the biological datasets, we have carried out inductive learning tasks on some of the common citation datasets, i.e. Cora, Citeseer [56] and Pubmed [47]. We follow [57] in carrying out the experiment, and use the largest connected component for each dataset with 20 training examples per class, 30 validation examples per class, and the rest of the data used for testing. The hyperparameters of our models and optimization procedure were based on the settings of the GAT model in [57, Table 4], which we have slightly fine-tuned on Cora and used for the other datasets. In particular, a single layer of `G2TN` or `G2T(A)N` is used with 64 functionals, max. tensor degree 2 and random walk length 5. The dropout rate was tuned to 0.9, while the attentional dropout was set to 0.3 in `G2T(A)N` . Optimization is carried out with Adam [32], a fixed learning rate of 0.01 and ℓ_2 regularization strength 0.01. Training is stopped once the validation loss does not improve for 50 epochs, and restored to the best checkpoint. For both of our models, `NoDiff` is used that we found to improve on the results as opposed to using increments of node features. The dropout rate had to be tuned as high as 0.9, which suggests very strong overfitting, hence the additional complexity of `AlgOpt` was also contrabeneficial, and `NoAlgOpt` was used. As such, each model employs a single hidden layer, which is followed by layer normalization that was found to perform slightly better than other normalizations, e.g. graph-level normalization.

The results of our models over 100 runs are reported in Table 4 compared with the 4 consistently best performing baselines on these datasets from [57], i.e. GCN [33], GAT [65], MoNet [45], and GraphSage [29] with a mean aggregator. Firstly on Cora, both `G2TN` and `G2T(A)N` outperform the baselines with a more significant improvement for `G2TN` . For CiteSeer, our models are left somewhat behind compared to the baselines in terms of accuracy. Finally, they are again competitive on Pubmed, where `G2TN` takes the top score with a very slight lead. Two consistent observations are: (1) `G2TN` and `G2T(A)N` have a lower variance than all baselines, (2) `G2TN` consistently outperforms `G2T(A)N` . The latter may be attributed to the observation that due to the severe overfitting on these datasets, the additional complexity of the attention mechanism in `G2T(A)N` is unhelpful for generalization.

K-hop Sanitized Splits. Recent work [50] has demonstrated that it is possible to make the previously considered citation datasets more suitable for testing the ability of a model to learn long-range information by dropping node labels in a structured way. Concretely, they use a label resampling strategy to guarantee that if a node is selected for a data split, none of its k-th degree neighbours are included in any splits, i.e. training, validation nor testing, allowing to reduce the effect of short-range "label imprinting". In practice, we select a maximal independent set from the graph with respect to the k-th power of the adjacency matrix with self-loops, and repeat the previous experiment with the same data splitting method, model choice and training settings as before. In this case, the experiment seed is also used to control the random maximal independent set that is selected.

The results of our models trained on the citation datasets sanitized this way are available in Table 5 computed over 100 seeds for $k = 1, 2$. As baseline results, we compare against the 5 best performing models from [50], where various GNN depths were also considered for each model, and we use the *best* reported result for each of the baselines. Overall, we can observe that all baseline models exhibit a very sharp drop in performance as k is increased, while for `G2TN` and `G2T(A)N` , the performance

Table 5: Accuracies of our models on k-hop sanitized citation datasets computed over 100 seeds compared with the 5 consistently best performing models from [50].

k	Dataset	GCN	GAT	g-U-net	HGNet-EP	HGNet-L	`G2TN` (ours)	`G2T(A)N` (ours)
1	Cora	76.7	78.5	78.1	77.2	77.1	$\mathbf{81.8 \pm 1.3}$	80.9 ± 1.4
	Citeseer	64.2	66.4	63.0	64.3	64.10	$\mathbf{68.1 \pm 1.3}$	66.6 ± 1.4
	Pubmed	75.8	75.9	75.8	77.0	76.3	$\mathbf{78.7 \pm 1.9}$	77.5 ± 2.0
2	Cora	72.0	73.4	74.4	74.0	75.4	$\mathbf{79.4 \pm 2.8}$	78.3 ± 2.9
	Citeseer	58.3	59.4	57.3	57.8	59.9	$\mathbf{63.6 \pm 1.8}$	62.2 ± 1.9
	Pubmed	72.1	73.1	72.4	72.9	75.1	$\mathbf{77.1 \pm 1.7}$	76.3 ± 1.6

Table 6: Computation time of one forward pass of one `G2TN` layer in milliseconds on a Nvidia GeForce 2080TI GPU for a graph with varying nodes (N) and edges (E), while node feature dimension is fixed at 128.

	$E = 5000$	$E = 10000$	$E = 20000$	$E = 40000$	$E = 80000$
$N = 500$	37	61	98	151	293
$N = 1000$	37	62	104	152	290
$N = 2000$	39	63	89	160	293
$N = 4000$	39	63	109	160	294
$N = 8000$	42	65	108	156	295

decrease is not nearly as pronounced. For both $k = 1$ and $k = 2$, our models perform better than the baselines on all datasets. This is explained by the fact that due to using random walks length of 5, the models can efficiently pick up on information outside of the sanitized neighbourhoods. As previously, `G2TN` performs better than `G2T(A)N` as the additional flexibility of the attention layer does not lead to improved generalization performance when severe overfitting is present. This experiment demonstrates that the proposed models efficiently pick up on long-range information within larger neighbourhoods, and it suggests that on inductive learning tasks they should be more robust to sparse labeling rates compared to common short-range GNN models.

H.4 Computation Time

Table 6 shows the computation time of one forward pass of a `G2TN` layer with various graphs. This empirically demonstrates that our layer does not depend on the number of nodes, and only depends linearly on the the number of edges as expected from the theoretical complexity.

Table 7 shows the computation time of one forward pass of a `G2TN` layer with a fixed graph and various model parameters. Empirically, our layer scales roughly linearly in both the walk length k and the maximum tensor level M. Linear complexity in walk length is expected, but the linearity in the maximum tensor level M is due to parallelization of the algorithm presented in Theorem 3. This is discussed in the pseudocode section of Appendix F.

H.5 Parameter Count

In both the NCI datasets, the `G2TN` and `G2T(A)N` have 505k and 519k trainable parameters respectively. Thus, our models are comparable in size to the GraphTrans, which is reported to have 500k trainable parameters [70].

H.6 Summary of Dataset Parameters

Here, we provide a summary of the dataset statistics and the model parameters used in the experiments. Here, M denotes the maximum tensor degree, k denotes the walk length, and R denotes the maximum rank of the low rank functions. Concretely, R is the number of node features used within the network, and is analogous to the width of a neural network. For the computation of the diameter of a disconnected graph, we take the largest diameter of any connected component.

Table 7: Computation time of one forward pass of one `G2TN` layer in milliseconds on a Nvidia GeForce 2080TI GPU for a fixed graph with $N = 2000$ nodes and $E = 10000$ edges. The walk length (k) and the maximum level M of the model are varied.

	$k = 2$	$k = 4$	$k = 6$	$k = 8$	$k = 10$
$M = 1$	14	26	45	59	70
$M = 2$	20	40	72	81	101
$M = 3$	26	55	95	115	163
$M = 4$	34	73	101	150	188
$M = 5$	42	92	139	187	237

Table 8: Dataset statistics and `G2TN` model parameters used.

	graphs	total nodes	total edges	avg. diam.	classes	layers	M	k	R
NCI1	4110	122k	132k	13.3	2	4	2	5	128
NCI109	4127	122k	132k	13.1	2	4	2	5	128
CORA	1	2485	5069	19	7	4	2	5	64
Citeseer	1	2110	3668	28	6	4	2	5	64
Pubmed	1	19k	44k	18	3	4	2	5	64