Processing large-scale Graphs with G-Signatures

Anonymous Authors[1]

Abstract

Graph neural networks (GNNs) have evolved into one of the most popular deep learning architectures. However, GNNs suffer from over-smoothing node information and, therefore, struggle to solve tasks where global graph properties are relevant. We introduce G-Signatures, a novel graph learning method that enables global graph propagation via randomized signatures. G-Signatures use a new graph conversion concept to embed graph structured information which can be interpreted as paths in latent space. We further introduce the idea of latent space path mapping. This allows us to iteratively traverse latent space paths, and, thus globally process information. G-Signatures excel at extracting and processing global graph properties, and effectively scale to large graph problems. Empirically, we confirm the advantages of G-Signatures at several classification and regression tasks.

1. Introduction

Graph neural networks (GNNs), like graph convolutional networks (Kipf & Welling, 2017), GraphSAGEs (Hamilton et al., 2017), graph attention networks (Veličković et al., 2018), or message passing GNNs (Gilmer et al., 2017) are one of the most popular and most successful deep learning architectures. GNNs are based on the principle of learning interactions between many entities in forward dynamics (Battaglia et al., 2018) and, therefore, advance physical simulations of molecular modeling, fluid dynamics, weather forecasting, and aerodynamics (Batatia et al., 2022; Li et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Mayr et al., 2021; Brandstetter et al., 2022b; Keisler, 2022; Lam et al., 2022). However, the multi-layer message aggregation scheme of GNNs is prone to over-smoothing node information even after a few propagation steps (Li et al., 2018; Chen et al., 2020a; Zhu et al., 2021; Alon & Yahav, 2021), yielding node representations that tend to be similar to each other. Furthermore, since GNNs usually process information as messages across edges, the number of messages grows exponentially with the width of a GNN's receptive field, resulting in over-squashing for more propagation steps (Zhu et al., 2020; Chen et al., 2020a; Alon & Yahav, 2021). Consequently, whole-graph classification and regression tasks that comprise long-range dependencies are very challenging for GNNs (Xu et al., 2018). There has been extensive work to mitigate effects of over-smoothing and over-squashing (Chen et al., 2020b; Hu et al., 2020; Liu et al., 2020b; Gu et al., 2020; Yang et al., 2021; Kim et al., 2021; Liu et al., 2022; Alon & Yahav, 2021; Rampášek et al., 2022). Prominent examples of such are Gated GCNs (Bresson & Laurent, 2017) or Graph Transformers (Dwivedi & Bresson, 2021), which are amongst the best performing methods for longer-range graph interactions (Dwivedi et al., 2022; Rusch et al., 2023a;b; 2022). However, all these GNN variants are still underperforming when predicting global properties of graphs.

In this work, we introduce G-Signatures, which efficiently learn randomized signatures to solve various graph tasks via gradient descent. Most notably, G-Signatures enable global graph propagation since at their core G-Signatures replace the local concept of message aggregation by globally traversing the graph. We achieve this by interpreting graphs as paths and calculate the signature thereof. Roughly speaking, a path is a mapping from an interval into a space where its *signature* describes the path uniquely in an efficient, computable way (Chen, 1954; 1957; 1958). However, the signature cannot be computed directly since it is an infinite sequence. *Truncated signatures* approximate this sequence by cutting at a specific level. G-Signatures use *randomized signatures*, which approximate truncated signatures via random feature mappings. Truncated signatures have been suggested as a layer in neural networks to represent input data (Kidger et al., 2019). Furthermore, randomized signatures have already been used for differential equations (Cuchiero et al., 2021), and for time series prediction (Compagnoni et al., 2022). Signatures have been used for (character) recognition tasks in machine learning (Yang et al., 2015; Li et al., 2017; Xie et al., 2018; Yang et al., 2017), as well as for data streams and pricing in finance

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review at ICML 2024 AI for Science workshop. Do not distribute.

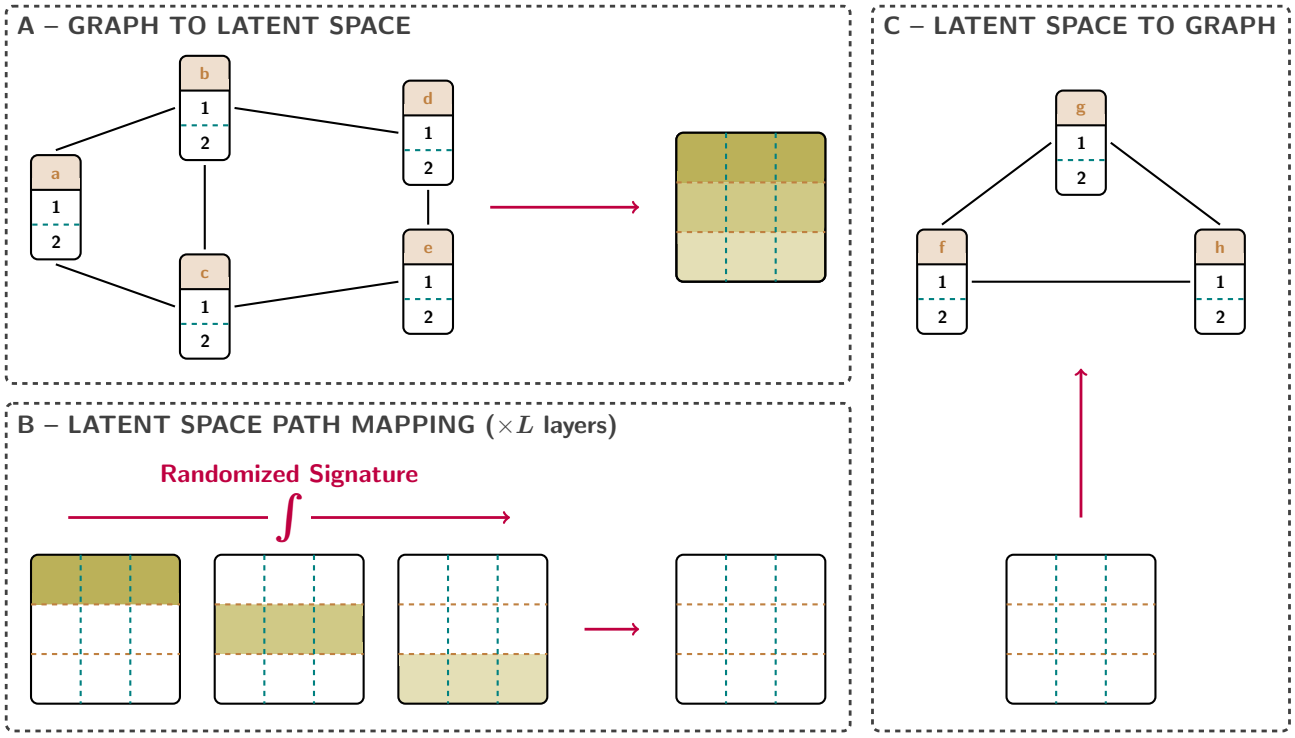

Figure 1. Schematic sketch of G-Signatures. G-Signatures apply randomized signature layers in latent space, which enables to globally and efficiently process graphs. *(A)* Graph structured data that is converted and interpreted as "path" along one dimension in latent space. *(B)* Latent space path mappings (LSPM) via randomized signature layers which traverse the converted graph iteratively on a per-feature basis. *(C)* The latent path is mapped to the target representation.

(Lyons et al., 2014; 2019; Kalsi et al., 2019), and for rough paths (Friz & Victoir, 2010; Lyons, 1998). Likewise, (Toth et al., 2022) model the local neighborhood of a given node in a GNN by interpreting it as a path over nodes with a limited path length, and summarize the path information with truncated signatures. In contrast to these approaches, we use randomized signatures to represent information as a path over the features of the nodes in a graph. G-Signatures learn how to refine the random mappings via gradient descent, that is, they learn to extract task-relevant information from the signature representation of a path. In doing so, we are, to the best of our knowledge, the first to present a framework for learning to extract signature information from graph structured data. In this work we introduce two novel concepts: (i) *graph conversion*, i.e., an efficient merging and embedding of graph information which we interpret as a path in latent space, and (ii) *latent space path mappings* (LSPM), i.e., new layers that map from one path to another in latent space. These two concepts constitute *G-Signatures*, which is a novel deep learning architecture with its own learning algorithm sketched in Figure 1. G-Signatures resemble GNNs, but in contrast to GNNs they offer an efficient and scalable solution to whole-graph classification and regression tasks. G-Signatures are a paradigm shift from collecting information locally for each node towards collecting information along paths through the graph. In G-Signatures, the signature transform is used in a similar way as Fourier analysis on Euclidean domains (Li et al., 2020) and, most notably, as spectral graph theory which can be seen as Fourier analysis on non-Euclidean domains (Bronstein et al., 2017; 2021). The largest disadvantage of spectral graph theory is that the eigendecomposition of the graph Laplacians is computationally expensive, especially for large graphs (Defferrard et al., 2016; Kipf & Welling, 2017). In contrast, G-Signatures allow for efficient layer-wise propagation of global graph information. To summarize, the contributions of this paper are:

- We introduce G-Signatures for global graph propagation based on randomized signatures.
- We establish the concepts of graph conversion and latent space path mappings (LSPM) which enable us to apply randomized signatures to graph structured data.
- In experiments, we demonstrate that G-Signatures (i) are able to learn global graph characteristics, (ii) offer an efficient and scalable solution to large graph problems, and (iii) are a more generally applicable method

not limited to typical GNN tasks.

2. Background and Related Work

In this section we discuss graph structured data, the randomized signature transform, and graph conversion.

Learning on graph structured data. We consider undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$, and edges $e_{ij} \in \mathcal{E}$, where d-dimensional node features $\boldsymbol{h}_i \in \mathbb{R}^d$ are attached to each of the nodes. Whether an edge between a pair of nodes (v_i, v_j) is contained in the graph $\mathcal{G}$ depends on the connectivity criterion between two nodes. For example, if distance is chosen as criterion, we might insert an edge when the cut-off radius $r_{\text{cut-off}}$ is below a threshold: $e_{ij} \in \mathcal{E} \iff d(v_i, v_j) \leqslant r_{\text{cut-off}}$. The connectivity is summarized in the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, which can be binary or weighted. Graph neural networks (GNNs) (Scarselli et al., 2009; Battaglia et al., 2018) are designed to learn from graph structured data and are by construction permutation equivariant with respect to the input.

Path. A *path* $X : [a, b] \to \mathbb{R}^d$ is a continuous mapping from an interval $[a, b]$ to $\mathbb{R}^d$ (Chevyrev & Kormilitzin, 2016). For a given path, the *signature* of the path summarizes its statistics (see Appendix A), whereas the mapping from a path to its signature is the *signature transform*.

Signature (transform) of a path. For a path $X : [a, b] \to \mathbb{R}^d$ with coordinate paths $X_t^1, \ldots, X_t^d$, where each $X^i[a, b] \to \mathbb{R}$ is a real-valued path, the signature transform as originally introduced by (Chen, 1954; 1957; 1958) is defined as:

$$S(X)_{a,t}^{i_1,\ldots,i_k} := \int_{a<t_k<t} \cdots \int_{a<t_1<t_2} \mathrm{d}X_{t_1}^{i_1} \ldots \mathrm{d}X_{t_k}^{i_k} , \quad (1)$$

where multi-index $i_1, \ldots, i_k \in \{1, \ldots, d\}^k$ and

$$S(X)_{a,t}^{i_1} := \int_{a<s<t} \mathrm{d}X_s^{i_1} = X_t^{i_1} - X_0^{i_1} \quad (2)$$

integrates each dimension separately, i.e., computes the per-coordinate increments. The pointwise evaluation $S(X)_{a,\cdot}^{i_1} : [a, b] \to \mathbb{R}$ is again a path. The collection of all feature increments concludes the first level of the signature. The second level is defined as:

$$S(X)_{a,t}^{i_1,i_2} := \int_{a<s<t} S(X)_{a,s}^{i_1} \, \mathrm{d}X_s^{i_2} \quad (3)$$

$$= \int_{a<r<s<t} \mathrm{d}X_r^{i_1} \, \mathrm{d}X_s^{i_2} , \quad (4)$$

with all further levels defined likewise. The double iterated integral of Equation (3) integrates one path, i.e., $S(X)^{i_1}$ against the other X^{i_2}. The collection of all iterated integrals of a path X in the interval $[a, b]$ is called the *signature* of the path X, and is denoted as $S(X)_{a,b}$ (Cuchiero et al., 2021):

$$S(X)_{a,b} := (1, S(X)_{a,b}^1, \ldots, S(X)_{a,b}^d, S(X)_{a,b}^{1,1}, \ldots, S(X)_{a,b}^{d,d}, \ldots) . \quad (5)$$

The signature itself exhibits a universal non-linearity property, that is linear functionals on the signature are dense in the set of functions on X (Arribas, 2018). An informal excerpt of this result is the following: A continuous function $f(X)$ of a path X can be approximated with an error lower or equal to ϵ by a linear mapping L of the signature $S(X)$:

$$\forall \epsilon > 0 \; \exists L : \| f(X) - L(S(X)) \| \leqslant \epsilon . \quad (6)$$

Hence, the signature of a path $X \in \mathbb{R}^d$ acts as a reservoir in terms of reservoir computing (Cuchiero et al., 2021), that is the signature acts as a fixed non-linear system and maps input signals into higher dimensional computational spaces. As a result, the signature gives a basis representation of functions on X in the Euclidean domain.

Truncated signature. The signature of Equation (5) itself cannot be computed, as it is an infinite sequence. Cutting this sequence at a specific level yields the truncated signature at level M:

$$S(X)_{a,b}^M := \left(1, S(X)_{a,b}^1, \ldots, S(X)_{a,b}^{d,\ldots,d}\right) , \quad (7)$$

where the superscript of the last entry is the multi-index denoting M-times the d-th coordinate of the path $X \in \mathbb{R}^d$ analog to Equation (1). In general, the number of signature terms at level M is

$$\left| S(X)_{a,b}^M \right| = (d^{M+1} - 1)/(d - 1) . \quad (8)$$

Randomized signature. The randomized signature is based on the Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss, 1984), which describes a distance-preserving, low-dimensional embedding of high-dimensional data. For a path $X : [0, T] \to \mathbb{R}^d$ where X consists of d coordinate paths sampled at N points $1 < \ldots < N = T$ (Compagnoni et al., 2022). The k-dimensional randomized signature $S_R(X)$ of X is the vector $\boldsymbol{z}_T \in \mathbb{R}^k$ which is obtained via N incremental updates. $\boldsymbol{z}_T$ approximates the signature of path X, with a vanishing error as $k \to \infty$. For more details see Theorem 3.3 of (Compagnoni et al., 2022). In the following, we refer to $\boldsymbol{z}_T$ and the respective $N-1$ preceding incremental update steps, i.e., $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T)^T$ as *randomized signature matrix* $\boldsymbol{Z} \in \mathbb{R}^{N \times k}$, which again is a path itself. A transposed and for G-Signatures modified computation can be seen in Algorithm 1 with a path $X : [0, T] \to \mathbb{R}^N$ sampled at d points $1 < \ldots < d = T$.

Algorithm 1 Randomized Signature Layer

Require: Matrix $\boldsymbol{X} \in \mathbb{R}^{d\times N}$ represents a path sampled at d points $1 < \ldots < d = T$, randomized signature size k, activation function σ, and number of signature heads p

Ensure: $\boldsymbol{z}_0 \in \mathbb{R}^k$, $\boldsymbol{A}_i \in \mathbb{R}^{k\cdot p\times k}$, $\boldsymbol{b}_i \in \mathbb{R}^{k\cdot p}$ from $\mathcal{N}(0, 1/k)$, and $\boldsymbol{W} \in \mathbb{R}^{k\times k\cdot p}$, $\boldsymbol{o} \in \mathbb{R}^k$ from $\mathcal{U}(-1/\sqrt{k\cdot p}, 1/\sqrt{k\cdot p})$.

1: **for** $j = 1$ **to** d **do**
2: $\quad \delta \boldsymbol{z}_j \leftarrow \boldsymbol{W}\big(\sum_{i=1}^{N} \sigma\big(\boldsymbol{A}_i \boldsymbol{z}_{j-1} + \boldsymbol{b}_i\big) X_j^i\big) + \boldsymbol{o}$
3: $\quad \boldsymbol{z}_j \leftarrow \boldsymbol{z}_{j-1} + \delta \boldsymbol{z}_j$
4: **end for**

The computation of the randomized signature can be seen as being part of the broad family of gated deep neural architectures, such as LSTMs (Hochreiter, 1991; Hochreiter & Schmidhuber, 1996; 1997), GRUs (Cho et al., 2014), or Highway Networks (Srivastava et al., 2015).

Randomized signature of graphs. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and node features $\boldsymbol{h}_i \in \mathbb{R}^d$ attached to them, we can interpret the graph as a path $G : [0, T] \to \mathbb{R}^N$ where G consists of N coordinate paths sampled at d points $1 < \ldots < d = T$. Put differently, time step j in the path consists of features $\{\boldsymbol{h}_{i,j}\}_{i=1}^N$ taken from all N nodes in the graph. The path is defined over features instead of nodes in order to keep favorable GNN properties like permutation invariance of processing node information. This also enables the processsing of global information of the entire graph in each step without depending on the concept of local information aggregation as in Message Passing GNNs (MPNNs). In Appendix C we give further intuition based on two important examples where the ability to propagate global graph information is vital to solve the given tasks. When interpreting a path as a random variable (Chevyrev & Kormilitzin, 2016; Chevyrev & Oberhauser, 2018) have shown that the signature extracts information about the statistical moments. More details can be found in Appendix A. This, paired with the signature's summarization and approximation capabilities (Chevyrev & Kormilitzin, 2016), makes it a natural candidate for path processing. The k-dimensional randomized signature $S_R(G)$ of G is the vector $\boldsymbol{z}_T \in \mathbb{R}^k$ which is obtained via d incremental updates. The randomized signature matrix of the graph $\boldsymbol{Z} \in \mathbb{R}^{d\times k}$ refers to the collection of d update steps leading to $\boldsymbol{z}_T$. By interpreting a graph with N nodes as a path with N coordinate paths as illustrated in Figure 2, we can process global feature information of the entire graph in each iterative step of Algorithm 1. Consequently, we have interpreted node information of any graph structured data as a path, and have shown how to calculate the randomized signature thereof. We define a procedure for edge embedding.

Edge embedding. Following (Roth et al., 2003; Duda et al., 2001; Schölkopf et al., 1998), we obtain an edge embedding via the eigendecomposition of the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$:

$$-\frac{1}{2}\boldsymbol{Q}\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\,, \tag{9}$$

where $\boldsymbol{Q} = \boldsymbol{I}_N - 1/N\,(1_1, \ldots, 1_N)^T\,(1_1, \ldots, 1_N)$ is used for normalization, and $\boldsymbol{I}_N$ is the $N \times N$ identity matrix, $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N)$ is a matrix with the eigenvectors as its columns, and $\boldsymbol{\Lambda} = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_N)$ is the diagonal matrix of the accompanying eigenvalues, sorted in decreasing order. The edge embedding matrix $\boldsymbol{E}_{\boldsymbol{A}} \in \mathbb{R}^{N\times m}$ is obtained by

$$\boldsymbol{E}_{\boldsymbol{A}} = \boldsymbol{V}_m\sqrt{\boldsymbol{\Lambda}_m}\,, \tag{10}$$

where $0 < m < N$ is the embedding dimension (hyperparameter), $\boldsymbol{\Lambda}_m \in \mathbb{R}^{m\times m}$ is the submatrix with the m largest eigenvalues on its diagonal and $\boldsymbol{V}_m \in \mathbb{R}^{N\times m}$ with the corresponding eigenvectors. For more details about the embedding procedure see Appendix B. Similar to the node information, we can now view $\boldsymbol{E}_{\boldsymbol{A}}^T \in \mathbb{R}^{m\times N}$ as N coordinate paths sampled at m points, and thus have also found a way to interpret edge information as a path. The edge embedding procedure also holds if $\boldsymbol{A}$ is replaced with an arbitrary dissimilarity matrix, e.g., when positional information is given as with point clouds, which is exemplified in Figure 3.

3. G-Signatures

We follow the Encode-Process-Decode framework of (Battaglia et al., 2018) and (Sanchez-Gonzalez et al., 2020). For the encoding, we introduce *graph conversion* which allows us to "convert" graph structured data to a latent representation. In latent space, we process the paths via randomized signature layers which traverse the converted graph on a per-feature basis by introducing the concept of *latent space path mappings* (LSPM). Finally, we decode the extracted information into a target representation.

Graph conversion. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and node features $\boldsymbol{h}_i \in \mathbb{R}^d$ attached to them, we can summarize the nodes to $\boldsymbol{G} \in \mathbb{R}^{d\times N}$, and the transposed edge embedding of Equation (10) to $\boldsymbol{E}_{\boldsymbol{A}}^T \in \mathbb{R}^{m\times N}$. Depending on the task at hand, we can either use $\boldsymbol{G}$, or $\boldsymbol{E}_{\boldsymbol{A}}^T$, or concatenate them in the first dimension. For the encoding, we sequentially map each dimension into a latent space such that we get a latent graph representation $\boldsymbol{X} \in \mathbb{R}^{h_1\times h_2}$ with hidden dimensions h_1 and h_2. This representation can be interpreted as a path $X : [0, T] \to \mathbb{R}^{h_2}$ sampled at h_1 points $1 < \ldots < h_1 = T$. The current graph conversion procedure is designed for homogeneous graphs, i.e., graphs with the same number of nodes and the same connectivity. We think

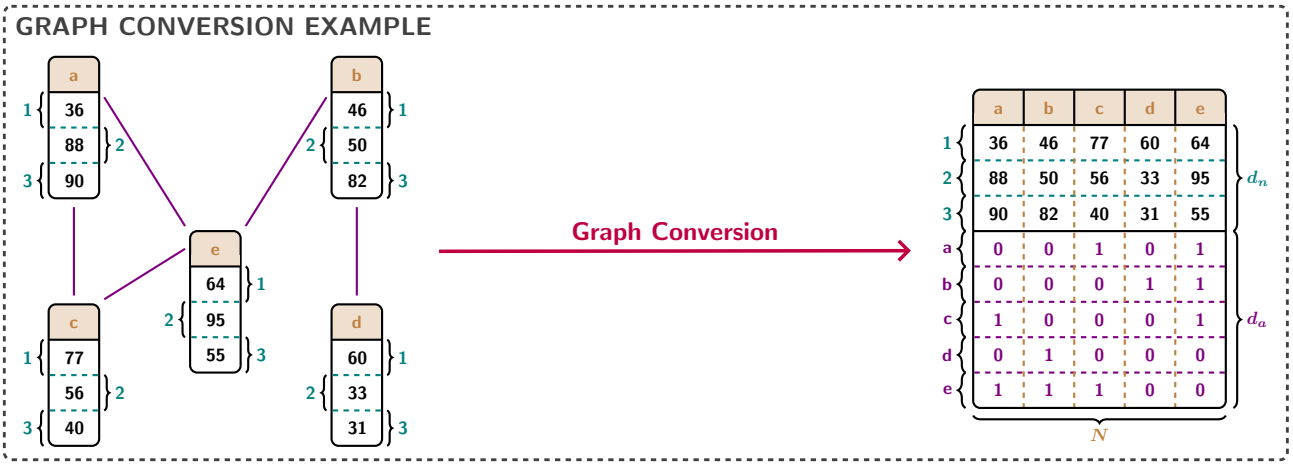

Figure 2. Example showcasing our proposed graph conversion concept. Graph conversion allows us to efficiently compress graph structured data into a latent representation while preserving node as well as adjacency information. For visual clarity the latent adjacency information is depicted in a simplified way where it is actually computed according to Equation (10).

that the graph conversion concept can be extended to heterogeneous graphs. Similarly, the edge embedding can be extended to edge features beyond scalar connectivity information. We will address these two extensions in future work. Both, graph conversion and our edge embedding strategy allow us to efficiently compress large-scale graphs without losing edge connectivity information. This in turn forms the basis for efficient propagation of graph information as evidenced in experiments where our method needs only a fraction of memory and runtime compared to baselines.

Latent space path mappings (LSPM). LSPM comprises a sequence of stacked randomized signature layers. More precisely, for $l \in \{1, \dots, L\}$ where L is the number of signature layers the input path to the l^{th} randomized signature layer $\boldsymbol{X}^l \in \mathbb{R}^{h_1 \times h_2}$ is mapped to an output path $\boldsymbol{X}^{l+1} \in \mathbb{R}^{h_1 \times h_2}$ in the same space. The mapping is based on the procedure as described in Algorithm 1, with the difference that $\boldsymbol{A_i}$, and $\boldsymbol{b}_i$ are learnable parameters. G-Signatures learn how to refine the random mappings via gradient descent, that is, they learn to extract task-relevant information from the signature representation of a path. This is done in a bidirectional manner along the axis that represents the time steps of the underlying path $1 < \ldots < h_1 = T$, resulting in the randomized signature matrix $\boldsymbol{Z}^l \in \mathbb{R}^{h_1 \times 2k}$ (again, $\boldsymbol{Z^l}$ is a path itself). We map the randomized signature matrix $\boldsymbol{Z}^l \in \mathbb{R}^{h_1 \times 2k}$ back to $\boldsymbol{X}' \in \mathbb{R}^{h_1 \times h_2}$, adding it to the input path via a residual connection: $\boldsymbol{X}^{l+1} = \boldsymbol{X}^l + \boldsymbol{X}'$. Consequently, we can stack several randomized signature layers, and thus enable mappings between paths in latent space. The randomized signature layers are easily parallelizable to multiple signature "heads" in similar vein to Multihead Attention (Vaswani et al., 2017). An overview of the LSPM procedure is visualized in the Appendix in Figure 9 with a more detailed sketch in Figure 8.

Characteristics of LSPM implementation. Since the naive application of the randomized signature does not produce desirable results we made important adjustments. In order to obtain an adequate learning behavior, three important adjustments to Algorithm 1 are required: (i) **sparsity** of the learnable weights $\boldsymbol{A}_i$, (ii) suitable weight **initialization**, and (iii) proper **activation** functions. Concerning (i), we experimentally found that dense parameters $\boldsymbol{A}_i$ dilute the signal $\boldsymbol{X}^l$ (similar to graph over-smoothing) leading to outputs with high noise-levels. We prevent the signal from diluting by sparsifying $\boldsymbol{A}_i$. Furthermore, we found that initializing the components of $\boldsymbol{A}_i, \boldsymbol{b}_i$ from $\mathcal{N}(0, 1)$ is prone to produce high weight values that lead to unfavorable learning dynamics. We, thus, reduce the weight magnitude by adaptively reducing the variance depending on the signature size k to $\mathcal{N}\left(0, 1/k\right)$. Finally, to prevent exploding signal values, we use the identity as activation function scaled by $1/h_2$. Exploding signal values arise due to the double summation of the bidirectional signature for large h_2 and non-vanishing X_j^i in Algorithm 1. The activation function compensates for large numbers of summation steps in case of a high-dimensional path space and implicitly reduces the variance of $\boldsymbol{A}_i, \boldsymbol{b}_i$ by a factor of $1/h_2^2$. We validate the importance of these adjustments by ablation studies shown in Appendix G.

Decoding. For the decoding, we use a similar approach as for the encoding, and sequentially map the latent space dimensions to the output dimensions.

4. Experiments

We test G-Signatures on several tasks, i.e., (i) global graph classification on the CoMA dataset (Ranjan et al., 2018), (ii) node regression on the Kardar–Parisi–Zhang (KPZ) (Kardar et al., 1986) equation with varying degrees of noise, and (iii) large graph edge regression on estimated time of arrival (ETA) tasks with varying sparsity degrees. For non-gridded graph structured data, i.e., CoMA and ETA tasks, we compare against Gated Graph Convolutional Networks (GGCNs) (Bresson & Laurent, 2017) and Graph Transformers (GTs) (Dwivedi & Bresson, 2021) since they are amongst the best performing methods in several long range graph benchmarks (Dwivedi et al., 2022). GGCNs (Bresson & Laurent, 2017) are a GCN (Kipf & Welling, 2017) based representative that uses learned edge gates to improve the aggregation procedure. GTs (Dwivedi & Bresson, 2021) generalize transformer networks on arbitrary graphs via node attention and pairwise modification of the attention scores via edge attributes. For the KPZ equation, we compare against ResNets (He et al., 2016) and Fourier Neural Operators (FNOs) (Li et al., 2020) on regularly-gridded data. We describe each task in detail, further information can be found in Appendix H.

4.1. Recognizing facial expressions

The convolutional mesh autoencode (CoMA) dataset (Ranjan et al., 2018) consists of 20465 graphs where each graph comprises 5023 nodes with 3 (positional) node features each, and 29990 edges, see Figure 3 for visualized datapoints. The task is to predict the facial expression for 12 different target labels. We use the 1483 samples test dataset as provided by the CoMA dataset implementation, and split the given training dataset into 17499 training and 1483 validation samples in a stratified way. For all methods, i.e.,

Figure 3. Exemplary samples in the CoMA dataset. Left: the original datapoints, right: their corresponding embedded versions, according to (Roth et al., 2003). Most relevant information of the original graph is visually contained in the converted graph representation.

G-Signatures, GGCNs, and GTs, we use the edge embedding of Equation (10) and set $m = 3$. This results in network inputs of $\mathbb{R}^{3\times 5023}$. GGCNs use 4 layers with a hidden dimension of 70, resulting overall in 104k parameters. GTs use 10 attention layers, with 8 attention heads each and a hidden dimension of 80, resulting in 913k parameters. For GGCNs, adjacency information is additionally provided to the graph convolution layers. G-Signatures benefit from the fact that we convert the graph to a latent space path of dimension $h_1 \times h_2 = 39 \times 39$, which substantially reduces the cost of memory and compute. G-Signatures outperform the baseline methods with an accuracy (evaluated on three replicates) of 93.74 ± 0.76, compared to 92.85 ± 0.42 for GTs, and 76.74 ± 1.19 for GGCNs. Most notably, G-Signatures are much less memory consumptive, and have much lower inference and training times as shown in Table 1.

4.2. Surrogate models for stochastic PDEs

Next, we show that our method can competitively perform with strong baselines on tasks it was not primarily designed for, making it a candidate for more general usage scenarios. More precisely, we stress-test the ability of G-Signatures to model spatially connected data, and, thus, force G-Signatures to not only traverse information on a per-feature basis, but also aggregate across nodes. Furthermore, using regularly gridded data, and comparing against state-of-the-art methods on those, i.e., FNOs (Liu et al., 2020a) and ResNets (He et al., 2016), presents a litmus test for each graph-tailored method. We, therefore, aim to learn surrogates on temporal evolving noisy data, that is from data obtained from numerical partial differential equation (PDE) solvers. Concretely, we look at the Kardar–Parisi–Zhang (KPZ) equation, which is a stochastic PDE that describes the spatial temporal change of $\boldsymbol{u}(x,t)$:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \nu \Delta \boldsymbol{u} + \frac{\lambda}{2}(\nabla \boldsymbol{u})^2 + \eta(x,t) \tag{11}$$

where ν and $\lambda \in \mathbb{R}$ are diffusion and viscosity coefficient, Δ and ∇ are Laplace- and Nabla-operator, and $\eta(x,t)$ is white Gaussian noise, find more information in Appendix D. The task can be interpreted as node regression task on an equidistant graph where the temporal inputs are the node features. We test for different noise levels, comparing against FNOs (Li et al., 2020), and ResNets (He et al., 2016), following the experimental setup presented in (Brandstetter et al., 2022a). Both architectures are built from translation equivariant layers, and are heavily used on equidistant grids. The input and output of the models are 10 timesteps in the channel dimension. The ResNet18 is a 1D variant of (He et al., 2016) with 4×2 residual blocks, and 128 hidden channels. Overall this results in 801k parameters. The FNO architecture uses 4 1D FNO layers as proposed in (Li et al., 2020) with 12 Fourier modes, resulting in 863k parameters. These layers have residual connections and are intertwined with GeLU (Hendrycks & Gimpel, 2016) non-linearities. G-Signatures convert the graph to a latent space representation $\boldsymbol{X} \in \mathbb{R}^{h_1 \times h_2}$ with $81 \leqslant h_1 = h_2 \leqslant 89$. We use one randomized signature layer for LSPM, overall resulting in an architecture with 53.7k parameters for the high noise dataset, 101k parameters for the low noise dataset, and 64.4k

Method	Accuracy	#Parameters[k]	#Layers	Memory[MiB]	Fwd.[s]	Fwd.+Bwd.[s]
GGCNs	76.74 ± 1.19	104	4	1,581	0.005	0.025
GTs	92.85 ± 0.42	913	10	1,875	0.052	0.101
G-Sigs.	93.74 ± 0.76	430	1	1,361	0.002	0.022

Table 1. Comparison of accuracy, parameter count, network depth, and memory/time consumption of compared methods on the CoMA dataset. Memory and time consumption is measured per datapoint.

parameters for the zero noise dataset. In Table 2 model performances are compared at different noise levels. See Figure 7 of the Appendix for more information. Although not tailored to regularly gridded data, G-Signatures are competitive with FNO and ResNet baselines, surpassing both baselines in the zero, and low noise settings. In the high noise setting, G-Signatures perform second best. This shows that G-Signatures keep up with strong baselines on regularly gridded tasks without using convolution operations, making it a more generally applicable method not limited to typical graph representation learning tasks.

4.3. Estimated time of arrival

The estimated time of arrival (ETA) is the time it is expected to take a vehicle, or a person, to arrive at a certain place (Derrow-Pinion et al., 2021; Hu et al., 2022; Wang et al., 2018). We model the ETA task by a graph with adjacency matrix $\boldsymbol{A}$. Each component A_{ij} represents the time it takes to get from node i to node j if the nodes are connected. On a map the nodes could for example represent intersections and only if a connecting street between nodes i and j exists, the value in the adjacency matrix will be the time it takes to get from i to j. Otherwise, we set the value of the edge connecting node i and j to infinity. The task is to predict the shortest travel time from node i to j by possibly traversing other nodes. By increasing the sparsity in the adjacency matrix, more nodes have to be traversed to get from node i to node j. We conducted experiments on graphs with 3 different node counts with 3 different connectivity levels (number of edges) each, resulting in 9 experimental settings, find more information in Appendix E. Gated GCNs and Graph Transformers additionally use connectivity information about the nodes as edge features for information aggregation across nodes. GGCNs use 10 layers with a hidden dimension of 70, resulting overall in 254k parameters. GTs use 8 attention layers, with 8 attention heads each and a hidden dimension of 32, resulting in 119k parameters. G-Signatures again benefit from the fact that we convert the graph to a latent space representation $\boldsymbol{X} \in \mathbb{R}^{h_1 \times h_2}$ with $32 \leqslant h_1 = h_2 \leqslant 95$. This substantially reduces the cost of memory and compute. The more the connectivity decreases the more important global aspects of the graph become since connecting two nodes can require propagating over many nodes in the graph. Each node-pair needs to be regressed as a scalar and we use MSE to measure the performance.

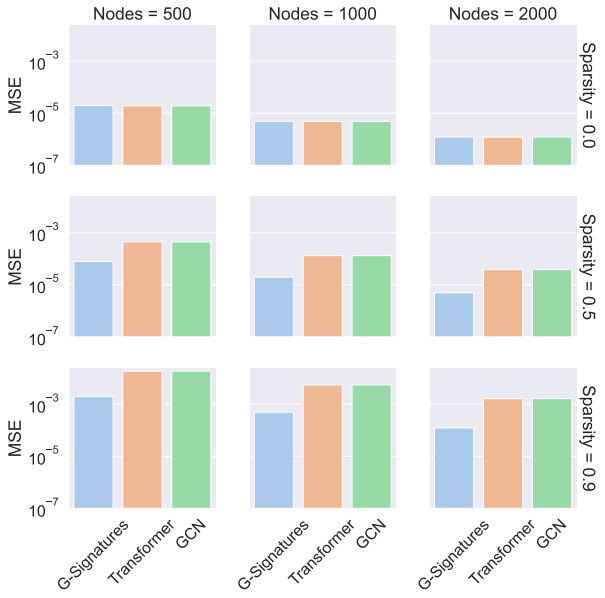

Figure 4. Results on ETA. When increasing the sparsity of the graphs, G-Signatures substantially outperform the competitors, often by an order of magnitude. See Table 4 of the Appendix for precise numerical results with error bounds.

Results are reported in Figure 4 and Table 4 in Appendix, confirming the advantage of global graph propagation via G-Signatures. All compared methods have very similar performance in case of high connectivity. This attributes to the fact that in these scenarios the propagation of global information is less important since all node pairs share an edge thus sufficient information can be processed locally. For example, GGCNs need relatively few steps to propagate information through the graph. However, when increasing the sparsity of the graphs, G-Signatures substantially outperform the competitors, often by an order of magnitude. GTs, to exchange information, are not restricted to the local neighborhood of each node at a given message passing iteration. Nevertheless, for ETA prediction information collected along the possible paths between distant source and sink nodes has to be propagated over many transit nodes what also exposes GTs to the risk of over-squashing. Whereas

Noise level	Standard deviation	ResNets[MSE]	FNOs[MSE]	G-Signatures[MSE]
High	5×10^{-3}	$4.501 \times 10^{-3} \pm 0.004 \times 10^{-3}$	$4.866 \times 10^{-3} \pm 0.022 \times 10^{-3}$	$4.774 \times 10^{-3} \pm 0.013 \times 10^{-3}$
Low	1×10^{-3}	$0.991 \times 10^{-3} \pm 0.018 \times 10^{-3}$	$1.197 \times 10^{-3} \pm 0.025 \times 10^{-3}$	$0.980 \times 10^{-3} \pm 0.007 \times 10^{-3}$
Zero	0×10^{-3}	$0.075 \times 10^{-3} \pm 0.006 \times 10^{-3}$	$0.160 \times 10^{-3} \pm 0.014 \times 10^{-3}$	$0.068 \times 10^{-3} \pm 0.002 \times 10^{-3}$

Table 2. Performance on the KPZ datasets with three different noise levels. The reported deviation is the standard error of the mean. The noise level is steered by the standard deviation of Gaussian noise η. G-Signatures benefit from lower noise levels.

#Nodes	#Edges	GGCNs	GTs	G-Sigs.
500	2.50×10^5	4,401	4,881	1,373
	1.25×10^5	2,901	3,205	1,383
	2.50×10^4	1,815	1,823	1,353
1,000	1.00×10^6	13,257	14,861	1,425
	5.00×10^5	7,355	9,399	1,417
	1.00×10^5	2,655	2,987	1,411
2,000	4.00×10^6	48,905	55,023	1,655
	2.00×10^6	39,756	46,165	1,635
	4.00×10^5	6,295	7,027	1,639

Table 3. Memory consumption of compared methods for processing one graph during training on all estimated time of arrival (ETA) datasets measured in MiB.

G-Signatures can easily propagate information on a global scale by using only a fraction of the memory consumption compared to state-of-the-art methods, summarized in Table 3.

5. Conclusion

We have introduced, G-Signatures, a novel learning paradigm for learning on graph structured data. G-Signatures are built around the concepts of graph conversion, which allows us to treat graph structured data as paths in latent space, and latent space path mapping (LSPM), which performs global graph propagation via randomized signature layers. Consequently, in contrast to conventional GNNs or Graph Transformers, G-Signatures excel at extracting global graph properties with substantially reduced memory and compute budget. We have further shown that G-Signatures are a more generally applicable method not limited to GNN-specific tasks.

Limitations and future work. In the current setting, G-Signatures are designed for homogeneous graphs, i.e., graphs with the same number of nodes and the same connectivity. For future work, we aim to extend G-Signatures towards heterogeneous graphs to provide applicability for a larger set of problems, e.g., molecular modeling. Furthermore, we plan to extend the edge embedding to edge features beyond scalar connectivity information. Finally, the current version traverses paths along one dimension, but we aim to traverse paths along concurrent dimensions, and thus combine information with a much denser and richer content.

References

Allamanis, M., Brockschmidt, M., and Khademi, M. Learning to represent programs with graphs. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJOFETxR-.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Arribas, I. P. Derivatives pricing using signature payoffs. *ArXiv*, 1809.09466, 2018.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv*, 1607.06450, 2016.

Batatia, I., Kovács, D. P., Simm, G. N. C., Ortner, C., and Csányi, G. MACE: higher order equivariant message passing neural networks for fast and accurate force fields. *ArXiv*, 2206.07697, 2022.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *ArXiv*, 1806.01261, 2018.

Brandstetter, J., Welling, M., and Worrall, D. E. Lie point symmetry data augmentation for neural pde solvers. *ArXiv*, 2202.07643, 2022a.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural PDE solvers. *ArXiv*, 2202.03376, 2022b.

Bresson, X. and Laurent, T. Residual gated graph convnets. *ArXiv*, 1711.07553, 2017.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, 2104.13478, 2021.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020a. doi: 10.1609/aaai.v34i04.5747.

Chen, K. T. Iterated integrals and exponential homomorphisms. *Proceedings of the London Mathematical Society*, s3-4(1):502–512, 1954. doi: 10.1112/plms/s3-4.1.502.

Chen, K. T. Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula. *Annals of Mathematics*, 65(1):163–178, 1957. doi: 10.1112/plms/s3-4.1.502.

Chen, K. T. Integration of paths–a faithful representation of paths by noncommutative formal power series. *Transactions of the American Mathematical Society*, 89(2): 395–407, 1958. doi: 10.2307/1993193.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 13–18 Jul 2020b.

Chevyrev, I. and Kormilitzin, A. A primer on the signature method in machine learning. *ArXiv*, 1603.03788, 2016.

Chevyrev, I. and Oberhauser, H. Signature moments to characterize laws of stochastic processes. *ArXiv*, 1810.10971, 2018.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. In *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*, volume 1809.09466, 2014.

Compagnoni, E. M., Biggio, L., Orvieto, A., Hofmann, T., and Teichmann, J. Randomized signature layers for signal extraction in time series data. *ArXiv*, 2201.00384, 2022.

Cuchiero, C., Gonon, L., Grigoryeva, L., Ortega, J.-P., and Teichmann, J. Expressive power of randomized signature. In *The Symbiosis of Deep Learning and Differential Equations*, 2021. URL https://openreview.net/forum?id=KWWFPULvmVw.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., Battaglia, P. W., Gupta, V., Li, A., Xu, Z., Sanchez-Gonzalez, A., Li, Y., and Velickovic, P. ETA prediction with graph neural networks in google maps. *ArXiv*, 2108.11482, 2021.

Duda, R. O., Hart, P. E., and Stork, D. G. Pattern classification 2nd ed. *John Willey & Sons Inc*, 2001.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *ArXiv*, 2012.09699, 2021. presented at AAAI 2021 Workshop on Deep Learning on Graphs: Methods and Applications.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *ArXiv*, 2003.00982, 2020.

Dwivedi, V. P., Rampásek, L., Galkin, M., A. Parviz, G. Wolf, A. T. L., and Beaini, D. Long range graph benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2022. Track Datasets and Benchmarks.

Floyd, R. W. Algorithm 97: Shortest path. *Communications of the ACM*, 5, 1962.

Friz, P. K. and Victoir, N. B. Multidimensional stochastic processes as rough paths: theory and applications. *Cambridge University Press*, 2010.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1263–1272. PMLR, 2017.

Gu, F., Chang, H., Zhu, W., Sojoudi, S., and Ghaoui, L. E. Implicit graph neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 11984–11995. Curran Associates, Inc., 2020.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In *Exploring network structure, dynamics, and function using NetworkX*, volume 7, pp. 11–15, 2008.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *ArXiv*, 1606.08415, 2016.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, 1207.0580, 2012.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. Advisor: J. Schmidhuber.

Hochreiter, S. and Schmidhuber, J. Bridging long time lags by weight guessing and "Long Short-Term Memory". In Silva, F. L., Principe, J. C., and Almeida, L. B. (eds.), *Spatiotemporal models in biological and artificial systems*, volume 37 of *Frontiers in Artificial Intelligence and Applications*, pp. 65–72. IOS Press, Amsterdam, Netherlands, 1996.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020.

Hu, X., Binaykiya, T., Frank, E., and Cirit, O. Deepreta: An eta post-processing system at scale. *ArXiv*, 2206.02127, 2022.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, 2015.

Johnson, W. and Lindenstrauss, L. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26, 1984.

Kalsi, J., Lyons, T., and Arribas, I. P. Optimal execution with rough path signatures. *ArXiv*, 1905.00728, 2019.

Kardar, M., Parisi, G., and Zhang, Y.-C. Dynamic scaling of growing interfaces. *Physical Review Letters*, 56(9):889, 1986.

Keisler, R. Forecasting global weather with graph neural networks. *ArXiv*, 2202.07575, 2022.

Kidger, P., Bonnier, P., Perez-Arribas, I., Salvi, C., and Lyons, T. Deep signature transforms. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information*

Processing Systems, volume 32. Curran Associates, Inc., 2019.

Kim, J., Oh, S., and Hong, S. Transformers generalize deepsets and can be extended to graphs & hypergraphs. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 28016–28028. Curran Associates, Inc., 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. Graphcast: Learning skillful medium-range global weather forecasting. *ArXiv*, 2212.12794, 2022.

Li, C., Zhang, X., and Jin, L. Lpsnet: a novel log path signature feature based hand gesture recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 631–639, 2017.

Li, Q., Han, Z., and Wu, X. Deeper insights into graph convolutional networks for semi-supervised learning. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 3538–3545. AAAI Press, 2018.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgbSn09Ym.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *ArXiv*, 2010.08895, 2020.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *ArXiv*, 2006.10108, 2020a.

Liu, J., Hooi, B., Kawaguchi, K., and Xiao, X. Mgnni: Multiscale graph neural networks with implicit layers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21358–21370. Curran Associates, Inc., 2022.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020b.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Lyons, T. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 2:215–310, 1998.

Lyons, T., Ni, H., and Oberhauser, H. A feature set for streams and an application to high-frequency financial tick data. In *International Conference on Big Data and Computing (ICBDC)*, 2014.

Lyons, T., Nejad, S., and Arribas, I. P. Model-free pricing and hedging in discrete time using rough path signatures. *ArXiv*, 1905.01720, 2019.

Mayr, A., Lehner, S., Mayrhofer, A., Kloss, C., Hochreiter, S., and Brandstetter, J. Boundary graph neural networks for 3d simulations. *ArXiv*, 2106.11299, 2021.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *In The Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4663–4673. PMLR, 2019.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515. Curran Associates, Inc., 2022.

Ranjan, A., Bolkart, T., Sanyal, S., and Black, M. J. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision (ECCV)*, pp. 725–741, 2018. URL http://coma.is.tue.mpg.de/.

Roth, V., Laub, J., Kawanabe, M., and Buhmann, J. M. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1551, 2003.

Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. Graph-coupled oscillator networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18888–18909. PMLR, 2022.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks. *ArXiv*, 2303.10993, 2023a.

Rusch, T. K., Chamberlain, B. P., Mahoney, M. W., Bronstein, M. M., and Mishra, S. Gradient gating for deep multi-rate learning on graphs. *ArXiv*, 2210.00513, 2023b.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 8459–8468. PMLR, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Schölkopf, B., Smola, A., and Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *ArXiv*, 1505.00387, 2015.

Toth, C., Lee, D., Hacker, C., and Oberhauser, H. Capturing graphs with hypo-elliptic diffusions. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 38803–38817. Curran Associates, Inc., 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

Veličković, P., Cucurull, G., asanova, C., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICRL)*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, D., Zhang, J., Cao, W., Li, J., and Zheng, Y. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Warshall, S. A theorem on boolean matrices. *Journal of the ACM*, 9:216–218, 1962.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 1968.

Xie, Z., Sun, Z., Jin, L., Ni, H., and Lyons, T. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition. *IEEE transactions on pattern analysisand machine intelligence*, 40:1903–1917, 2018.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5449–458. PMLR, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yang, W., Jin, L., and Liu, M. Chinese character-level writer identification using path signature feature, dropstroke and deep cnn. In *13th International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

Yang, W., Lyons, T., Ni, H., Schmid, C., Jin, L., and Chang, J. Leveraging the path signature for skeleton-based human action recognition. *ArXiv*, 1707.03993, 2017.

Yang, Y., Liu, T., Wang, Y., Zhou, J., Gan, Q., Wei, Z., Zhang, Z., Huang, Z., and Wipf, D. Graph neural networks inspired by classical iterative algorithms. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11773–11783. PMLR, 2021.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Zhu, D.-H., Dai, X.-Y., and Chen, J.-J. Pre-train and learn: Preserving global information for graph neural networks. *Journal of Computer Science and Technology*, 36(6): 1420–1430, 2021. doi: 10.1007/s11390-020-0142-x.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804. Curran Associates, Inc., 2020.

Zwicker, D. py-pde: A python package for solving partial differential equations. *Journal of Open Source Software*, 5(48):2158, 2020. doi: 10.21105/joss.02158. URL https://doi.org/10.21105/joss.02158.

A. Statistical moments from the signature

Interpreting the path X as random variable with a realization resulting in the dataset $\{X_i\}_{i=1}^N$ (Chevyrev & Kormilitzin, 2016) have shown that truncating the signature at level L determines statistical moments up to level L. This is exemplary demonstrated for the first two moments. Using the canonical cumulative lead-lag embedding of $\{X_i\}_{i=1}^N$ we get

$$
\begin{aligned}
S(X)^1 = S(X)^2 &= \sum_{i=1}^{N} X_i \\
S(X)^{1,1} = S(X)^{2,2} &= \frac{1}{2}\left(\sum_{i=1}^{N} X_i\right)^2 \\
S(X)^{1,2} &= \frac{1}{2}\left[\left(\sum_{i=1}^{N} X_i\right)^2 + \sum_{i=1}^{N} X_i^2\right] \\
S(X)^{2,1} &= \frac{1}{2}\left[\left(\sum_{i=1}^{N} X_i\right)^2 - \sum_{i=1}^{N} X_i^2\right].
\end{aligned}
$$

From this we get that

$$
\begin{aligned}
Mean(X) &= \frac{1}{N} S(X)^1 \\
Var(X) &= -\frac{N+1}{N^2} S(X)^{1,2} + \frac{N-1}{N^2} S(X)^{2,1}.
\end{aligned}
$$

For a 1-dimensional path X its corresponding signature is

$$
S(X) = \left(1, X_N - X_1, \frac{(X_N - X_1)^2}{2!}, \cdots, \frac{(X_N - X_1)^k}{k!}, \cdots\right).
$$

Similarly, interpreting X as stochastic process its expected signature is then

$$
\mathbb{E}[S(X)] = \left(1, \mathbb{E}[X_N - X_1], \frac{\mathbb{E}[(X_N - X_1)^2]}{2!}, \cdots, \frac{\mathbb{E}[(X_N - X_1)^k]}{k!}, \cdots\right)
$$

which motivates the question of how this moment-like sequence relates to the law of X. (Chevyrev & Oberhauser, 2018) have shown that under certain assumptions the expected signature of a path $X : [a, b] \to \mathbb{R}^d$ indeed characterizes its law such that it can be thought of as a generalization of the moment-generating function of a real-valued random variable.

B. Edge Embedding

Following (Roth et al., 2003; Duda et al., 2001; Schölkopf et al., 1998), edge information is included into the nodes via the eigendecomposition of a dissimilarity matrix $\boldsymbol{D} \in \mathbb{R}^{N \times N}$:

$$
\begin{aligned}
\boldsymbol{Q} &= \boldsymbol{I}_n - \frac{1}{n} (1_1, \ldots, 1_n)^T (1_1, \ldots, 1_n) \ , \\
\boldsymbol{S}^c &= -\frac{1}{2} \boldsymbol{Q} \boldsymbol{D} \boldsymbol{Q} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^T \ ,
\end{aligned}
\tag{12}
$$

This formula is derived from the following statements. Let $\boldsymbol{M} \in \{0, 1\}^{n \times k}$ be a binary stochastic assignment matrix with $\sum_{\nu=1}^{k} M_{i\nu} = 1$. The pairwise clustering cost function is used to derive meaningful embedding vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ from a corresponding dissimilarity matrix $\boldsymbol{D}$ with $D_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The pairwise clustering cost function is defined as:

$$
H^{pc} = \frac{1}{2} \sum_{\nu=1}^{k} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} M_{i\nu} M_{j\nu} D_{ij}}{\sum_{l=1}^{n} M_{l\nu}} \ ,
\tag{13}
$$

where $\boldsymbol{D}$ is an arbitrary dissimilarity matrix between embedding vectors $\{\boldsymbol{x}_i\}_{i=1}^n$. Assuming $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^2$ as the dissimilarity function of choice the pairwise clustering cost function H^{pc} is equal to the k-means cost function iff $\boldsymbol{S}^c$ as described in Equation (12) is positive semidefinite (Duda et al., 2001; Roth et al., 2003). Minimizing H^{pc} leads to an optimal assignment matrix $\boldsymbol{M}$. For possibly indefinite $\boldsymbol{S}^c$, the given dissimilarity matrix $\boldsymbol{D}$ needs to be centralized and shifted with and $\boldsymbol{e}_n = (1_1, \ldots, 1_n)$. A diagonal shift of Equation (12) corresponds to an off-diagonal shift of the dissimilarity matrix $\boldsymbol{D}$ with the smallest eigenvalue of $\lambda(\boldsymbol{S}^c)$:

$$\tilde{\boldsymbol{D}} = \boldsymbol{D} - 2\lambda_n\left(\boldsymbol{S}^c\right)\left(\boldsymbol{e}_n^T\boldsymbol{e}_n - \boldsymbol{I}_n\right) \ . \tag{14}$$

Substituting D_{ij} with $\tilde{D}_{ij}$ in Equation (13) gives us the k-means cost function. Equivalently to Equation (12), the centralized squared Euclidean scoring matrix is

$$\tilde{\boldsymbol{S}}^c = -\frac{1}{2}\tilde{\boldsymbol{D}}^c = -\frac{1}{2}\boldsymbol{Q}\tilde{\boldsymbol{D}}\boldsymbol{Q} \ . \tag{15}$$

Following the definition of the kernel principal component analysis (PCA), the original embedding vectors can be reconstructed by an eigendecomposition of Equation (16) (Schölkopf et al., 1998; Roth et al., 2003):

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{I}_n - \frac{1}{n}\left(1_1, \ldots, 1_n\right)^T\left(1_1, \ldots, 1_n\right) \ , \\ \boldsymbol{S} &= -\frac{1}{2}\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\boldsymbol{T}} \ , \end{aligned} \tag{16}$$

where $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ is a matrix with the eigenvectors as its columns, and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of the corresponding eigenvalues, sorted in decreasing order. The embedding vectors are recovered by

$$\boldsymbol{X} = \boldsymbol{V}_m\left(\boldsymbol{\Lambda}_{\boldsymbol{m}}\right)^{0.5} \ , \tag{17}$$

where $0 < m < n$ is the embedding dimension, $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, $\boldsymbol{\Lambda}_m \in \mathbb{R}^{m\times m}$ is the submatrix with the m largest eigenvalues on its diagonal and $\boldsymbol{V}_m \in \mathbb{R}^{N\times m}$ with the corresponding eigenvectors.

C. G-Signatures can propagate global graph information in each step

. Global positional encoding

Prediction tasks on peptides molecular datasets as in (Dwivedi et al., 2022) require a model to capture long-range interactions among atoms. For a given peptide they contain the peptide sequence, molecular graph, function, and 3D structure of the peptide. However, the graphs used for modeling the peptides correspond to 1D amino acid chains that do not have 2D or 3D peptide structure information included. This leads to large graphs with approx. 150 nodes each, and makes it important for the used ML model to identify the location of an amino acid in the graph. To enable this, a common candidate solution is global positional encoding (Dwivedi et al., 2020) for each node.

To illustrate this more precisely we use the following simplified examples from (Rampášek et al., 2022). One of the tasks is to differentiate between two Circular Skip Link (CSL) graphs (Murphy et al., 2019) as in the first two columns of Figure 5. In the first row thereof it can be seen that the 1-Weisfeiler-Lehman test (1-WL) (Weisfeiler & Leman, 1968) produces the same coloring of all nodes in the two non-isomorphic graphs. This means that the 1-WL cannot differentiate between them. This also holds for MPNNs since their expressive power is upper-bounded by 1-WL (Xu et al., 2019; Morris et al., 2019). This problem can be solved by using a global positioning encoding as shown in the second row. Similarly, in the third column for a Decalin molecular graph which has two rings of all Carbon atoms 1-WL (and thus MPNNs) would generate the same color for nodes a, b, and c, d, respectively. Thus, for link prediction the potential links (a, d) and (b, d) would be indistinguishable. Again, this issue can be solved by global positional encoding.

However, for large graphs with many nodes MPNNs miss to propagate such global structural information between distant nodes. This is caused by the well-known over-squashing phenomenon which is due to the local information aggregation mechanism for node updates. This means that crucial location information cannot be propagated by MPNNs. On the other hand, G-Signatures have access to this information as they do not rely on message passing and process the information for a given feature (and thus positional encoding) of all nodes at once.

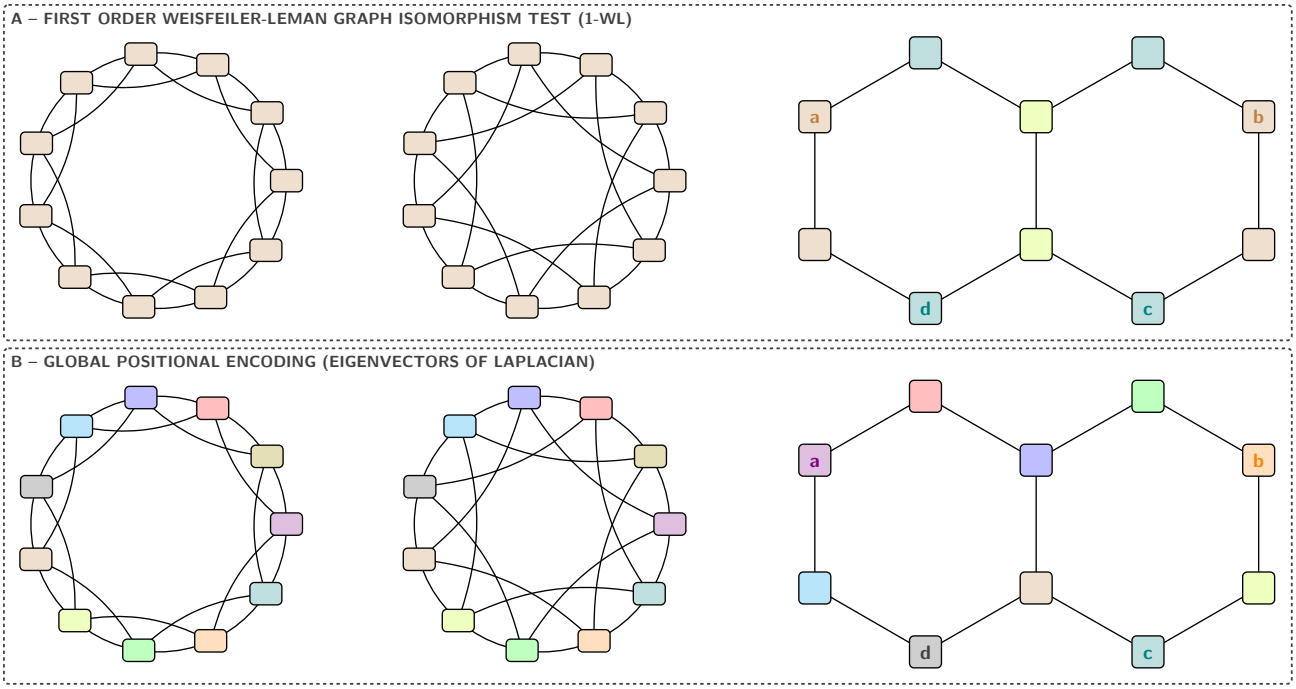

Figure 5. Showcasing the importance of global positional encoding for graph structured problems. *(A)* In the left two columns the 1-WL incorrectly identifies the two graphs as being isomorphic. Similarly, it cannot distinguish for example node pairs (a,d) and (b,d) in the right column. *(B)* When using a global positional encoding on the other hand these graphs and node pairs can be distinguished. Figure is a modified version of Figure C.1 in (Rampášek et al., 2022).

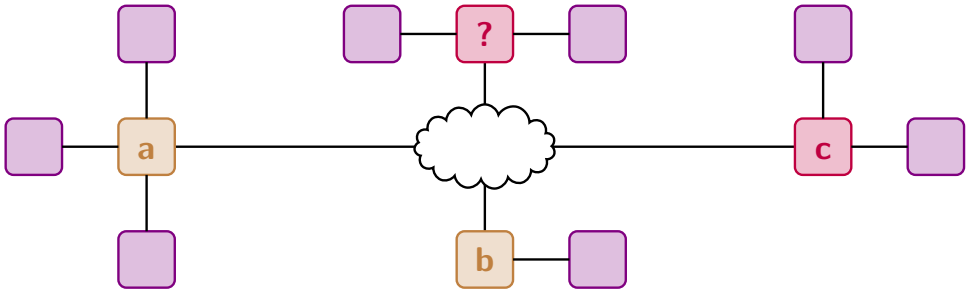

Figure 6. Showcasing the importance of global information propagation at the example of the neighbors match problem. To solve the task neighborhood information of distant nodes has to be propagated to the node marked with a question mark which can easily lead to over-squashing. Figure is a modified version of Figure 2 in (Alon & Yahav, 2021).

. Neighbors Match Problem

Another task where global graph propagation plays an important role is for real-world problems where computer program code is modeled as a graph (Allamanis et al., 2018). For example, long-range dependencies due to the usage of the same variable in distant locations are considered in this context. One such application would be the detection of variable misuses based for example on the variable types and values.

A simplified abstraction of this task is the Neighbors Match problem from (Alon & Yahav, 2021). In the example illustrated in Figure 6 the goal is to predict the correct label of the target node marked with a question mark. In this case the correct label is ”c” since the node with label ”c” has the same number of neighbors as the node in question. For large graphs this task can easily lead to over-squashing in MPNNs, since the information from all nodes has to be propagated to the target node, whereas G-Signatures can immediately process this information for the entire graph.

D. Surrogate Models for Stochastic PDEs

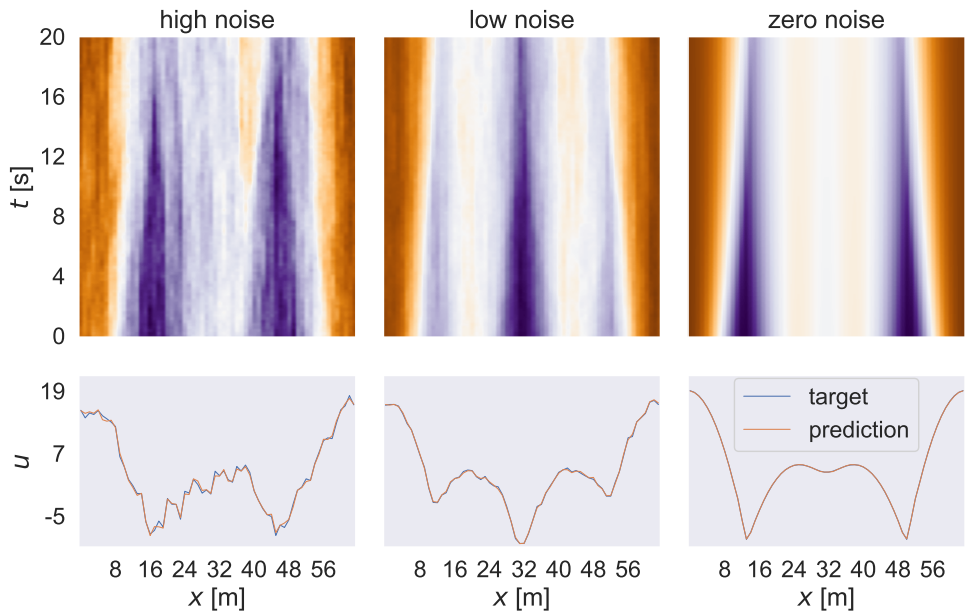

Figure 7. First row: Visualization of the dynamics of the KPZ equation unrolled over time. Second row: Exemplary comparison of G-Signatures predictions to target values at a fixed timestep of the above trajectory.

The term $\frac{\lambda}{2}(\nabla \boldsymbol{u})^2$ in Equation (11) is responsible for modeling shock waves. In one spatial dimension the KPZ equation corresponds to a stochastic version of the viscous Burgers' equation – a widely used PDE which occurs e.g., in fluid mechanics, or traffic modeling. Similarly, larger values for ν have a smoothing effect, and both smaller values for ν and larger values for λ result in larger shock formations. The noise component $\eta(x,t)$ presents an additional difficulty in modeling the dynamics of Equation (11). We aim to learn the mapping of $\boldsymbol{u}(x,t)$ to later points in time, We consider a modified implementation of the *py-pde* package (Zwicker, 2020), obtaining data on $n_x = 64$ spatial grid with temporal resolution of $\Delta t = 0.2$. The data is split into 512 training trajectories, 128 validation trajectories, and 128 test trajectories. Each time point of the respective training trajectories is used as individual starting point to obtain one training sample. The task is to predict the dynamics 10 timesteps into the future given 10 input timesteps, where performance is measured using the mean squared error (MSE).

E. Estimated Time of Arrival

We use the Floyd-Warshall (Floyd, 1962; Warshall, 1962) algorithm of the Python package *NetworkX* (Hagberg et al., 2008) to obtain ground truth values for the travel time along the shortest path between two nodes. Each node has 3 features and we use the edge embedding of Equation (10) setting $m = 3$ such that the network inputs for e.g. a graph with 500 nodes are in $\mathbb{R}^{3\times 500}$, which is the input to G-Signatures. We split each dataset in a random way, such that 512 graphs contribute to the training set, 128 graphs to the validation set, and 128 graphs to the test set. During training, each graph of the training set is augmented by applying a random permutation with respect to its nodes. This increases the potential total count of differently arranged training graphs by a factor of $\{500!, 1000!, 2000!\}$ for the respective ETA dataset.

#Nodes	#Edges	GGCNs[MSE]	GTs[MSE]	G-Signatures[MSE]
500	2.50×10^5	$1.936 \times 10^{-5} \pm 0.242 \times 10^{-8}$	$1.951 \times 10^{-5} \pm 9.466 \times 10^{-8}$	$1.979 \times 10^{-5} \pm 0.523 \times 10^{-8}$
	1.25×10^5	$4.444 \times 10^{-4} \pm 0.027 \times 10^{-7}$	$4.444 \times 10^{-4} \pm 0.237 \times 10^{-7}$	$0.785 \times 10^{-4} \pm 0.421 \times 10^{-7}$
	2.50×10^4	$1.846 \times 10^{-2} \pm 0.171 \times 10^{-5}$	$1.846 \times 10^{-2} \pm 0.177 \times 10^{-5}$	$0.196 \times 10^{-2} \pm 0.067 \times 10^{-5}$
1,000	1.00×10^6	$0.490 \times 10^{-5} \pm 0.983 \times 10^{-8}$	$0.489 \times 10^{-5} \pm 1.108 \times 10^{-8}$	$0.493 \times 10^{-5} \pm 0.206 \times 10^{-8}$
	5.00×10^5	$1.317 \times 10^{-4} \pm 0.304 \times 10^{-7}$	$1.317 \times 10^{-4} \pm 0.168 \times 10^{-7}$	$0.197 \times 10^{-4} \pm 0.024 \times 10^{-7}$
	1.00×10^5	$0.548 \times 10^{-2} \pm 0.069 \times 10^{-5}$	$0.548 \times 10^{-2} \pm 0.001 \times 10^{-5}$	$0.049 \times 10^{-2} \pm 0.008 \times 10^{-5}$
2,000	4.00×10^6	$0.123 \times 10^{-5} \pm 0.229 \times 10^{-8}$	$0.126 \times 10^{-5} \pm 2.221 \times 10^{-8}$	$0.124 \times 10^{-5} \pm 0.016 \times 10^{-8}$
	2.00×10^6	$0.383 \times 10^{-4} \pm 0.014 \times 10^{-7}$	$0.383 \times 10^{-4} \pm 0.094 \times 10^{-7}$	$0.049 \times 10^{-4} \pm 0.010 \times 10^{-7}$
	4.00×10^5	$0.163 \times 10^{-2} \pm 0.018 \times 10^{-5}$	$0.163 \times 10^{-2} \pm 0.003 \times 10^{-5}$	$0.012 \times 10^{-2} \pm 0.001 \times 10^{-5}$

Table 4. Results on the estimated time of arrival (ETA) datasets. The performance is reported by the mean squared error (MSE) with the standard error of the mean as the deviation. G-Signatures, Gated GCNs, and Graph Transformers are compared for different graph sizes, and different sparsity levels. When increasing the sparsity (reducing the number of edges) of the graphs, G-Signatures substantially outperform the competitors, often by an order of magnitude.

F. Latent Space Path Mapping

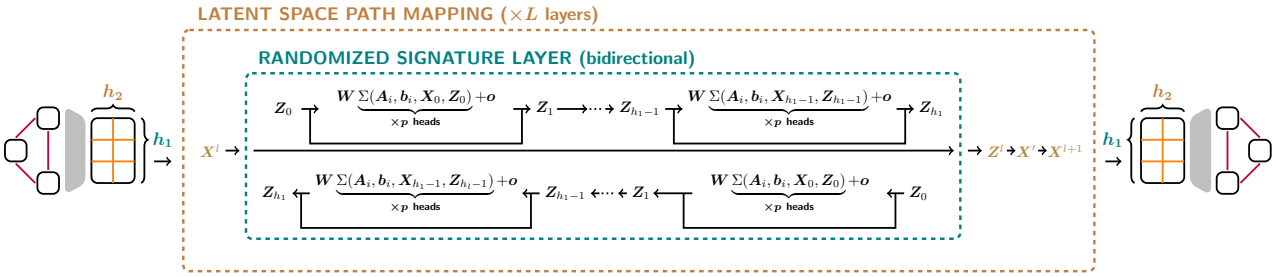

Figure 8. Illustration of the G-Signatures architecture. Graph conversion allows us to map graph structured data into a path $X : [0, T] \to \mathbb{R}^{h_2}$ in the latent space sampled at h_1 points. This constitutes the latent space representation of a graph. Latent space path mappings (LSPM) process this latent representation sequentially via stacked randomized signature layers. The decoder maps the latent graph representation to a task dependent output space.

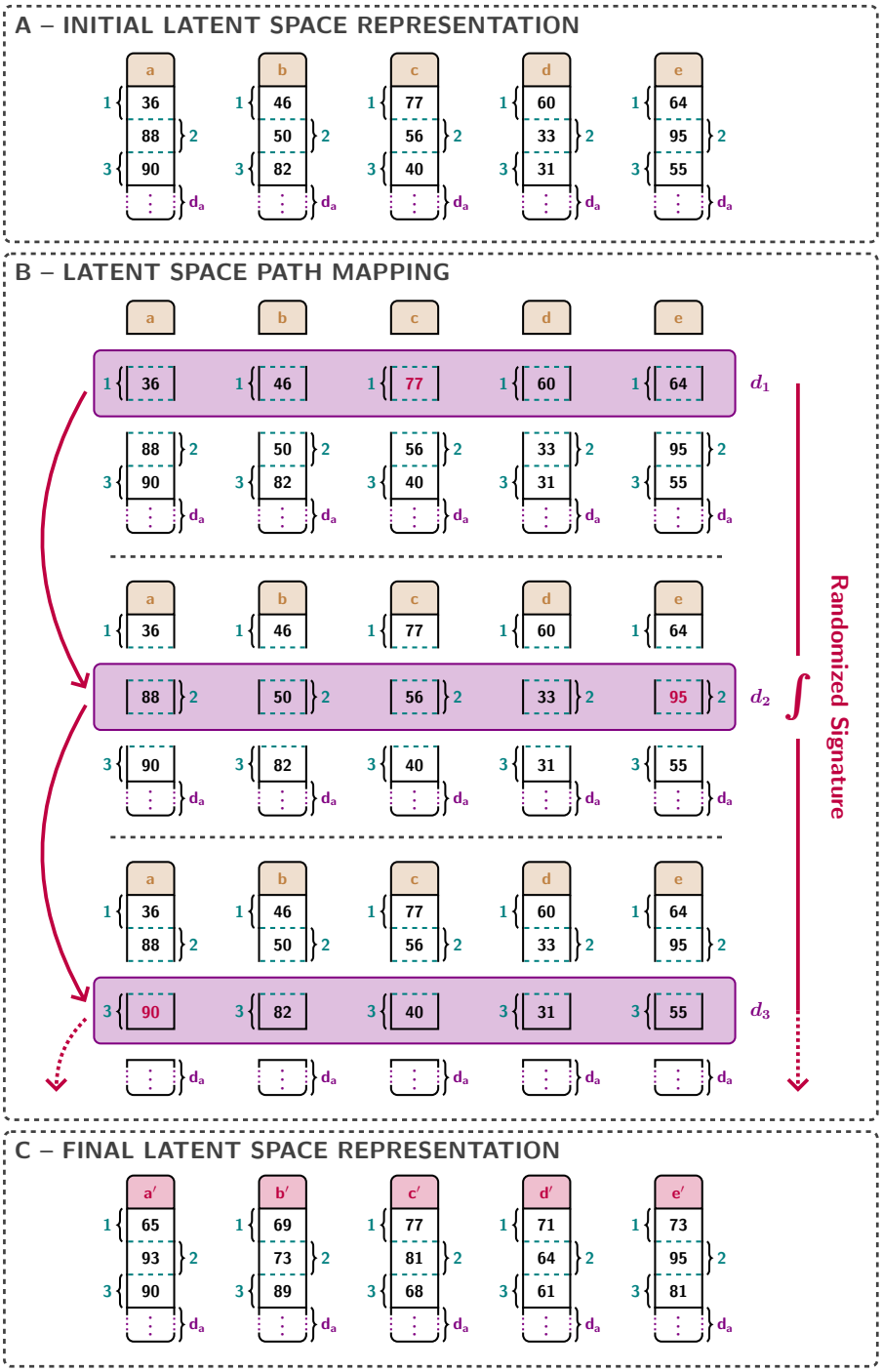

Figure 9. Latent space path mappings (LSPMs) traverse the initial latent space representation of a graph iteratively on a per-feature basis by applying randomized signature layers. In each iteration the randomized signature layers access global information of the entire latent graph in a parallel fashion, thus mitigating any negative smoothing and squashing effects. Exemplary, we showcase finding the maximum feature value of all latent nodes in purple. As our method operates on all latent nodes in parallel, the maximum value per feature is found in a single step.

G. Ablation Studies

We compare results of G-Signatures against ablated versions thereof on the KPZ dataset with a low noise level. We investigate the effect of all combinations of our adjustments. We apply the same hyperparameter search procedure as for the main experiments of G-Signatures. Each entry in Table 5 shows the corresponding result by omitting a certain combination of adjustments. The diagonal elements of both sub-tables indicate the omission of the respective row/column element, e.g., a combination of SPARSITY and SPARSITY corresponds to the omission of just the sparsity adjustment, a combination of SPARSITY and INITIALIZATION corresponds to the omission of the sparsity, as well as the initialization adjustments. The "$\cdots$" serves as a placeholder since the lower sub-table is symmetric, and "—" indicates that the given combination is not applicable. The TRAINABILITY ablation refers to a fixed randomized signature, i.e. A_i and b_i are kept fixed. It can be clearly seen that each adjustment is necessary to improve the performance of G-Signatures, and that omitting them steadily worsens the results for all of their combinations.

Ablations	NONE (OURS)	TRAINABILITY	ALL (ORIGINAL)
None (ours)	$0.980 \times 10^{-3} \pm 0.007 \times 10^{-3}$	—	—
Trainability	—	$1.022 \times 10^{-3} \pm 0.005 \times 10^{-3}$	—
All (original)	—	—	$1.308 \times 10^{-3} \pm 0.018 \times 10^{-3}$

Ablations	SPARSITY	INITIALIZATION	ACTIVATION
Sparsity	$0.983 \times 10^{-3} \pm 0.001 \times 10^{-3}$	$\cdots$	$\cdots$
Initialization	$0.994 \times 10^{-3} \pm 0.008 \times 10^{-3}$	$0.987 \times 10^{-3} \pm 0.007 \times 10^{-3}$	$\cdots$
Activation	$0.996 \times 10^{-3} \pm 0.009 \times 10^{-3}$	$1.003 \times 10^{-3} \pm 0.005 \times 10^{-3}$	$0.990 \times 10^{-3} \pm 0.005 \times 10^{-3}$
Sigmoid	—	—	$0.993 \times 10^{-3} \pm 0.004 \times 10^{-3}$
TanH	—	—	$0.998 \times 10^{-3} \pm 0.004 \times 10^{-3}$

Table 5. Performance results of G-Signatures compared to its ablated versions on the KPZ dataset with a low noise level. The reported deviation is the standard error of the mean. G-Signatures perform best with all LSPM adjustments applied.

It is notable that when omitting the sparsity adjustment, i.e., using dense matrices for A_i, both the performance decreases and the method is much less memory- and compute-efficient. More precisely, there is a quadratic instead of a linear scaling for A_i, as can be seen in Figure 10.

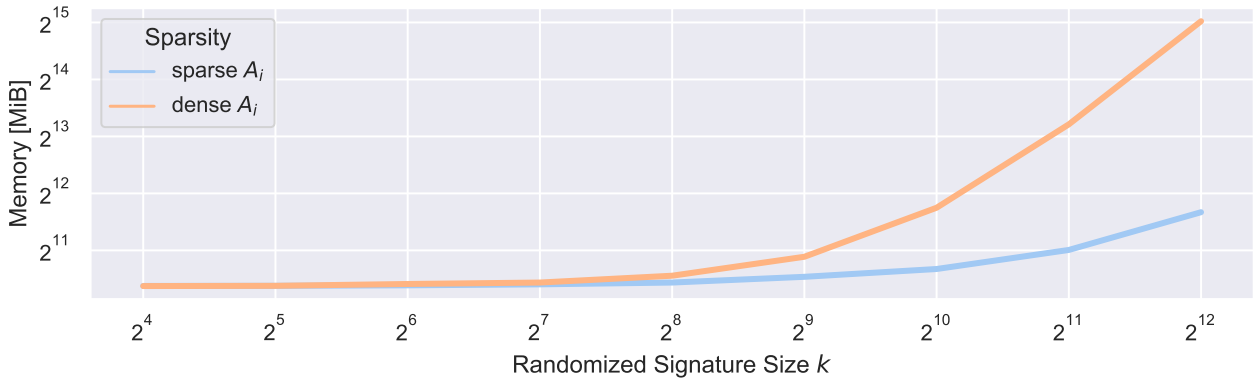

Figure 10. Memory consumption in MiB of G-Signatures on an NVIDIA A100 GPU 40GB for a single sample during training. All hyperparameters are kept fixed, except size k of A_i and b_i of the randomized signature layer (see Algorithm 1 for details).

Additionally, many hyperparameter settings of the ablated versions cannot even be trained successfully. We illustrate this in more detail starting from Figure 11 to Figure 18. For this, we investigate for all ablations in Table 5 the magnitude of the values computed by the signature based on Algorithm 1. In each of the figures the absolute values per timestep and signature size (averaged over samples) are plotted in the lower-right sub-figure. The range of these values across the signature sizes

per step is visualized in the upper-right plot. The range across the steps per signature size is shown in the left sub-figure. It can be seen that in most cases the computation leads to exploding values after a few steps already.

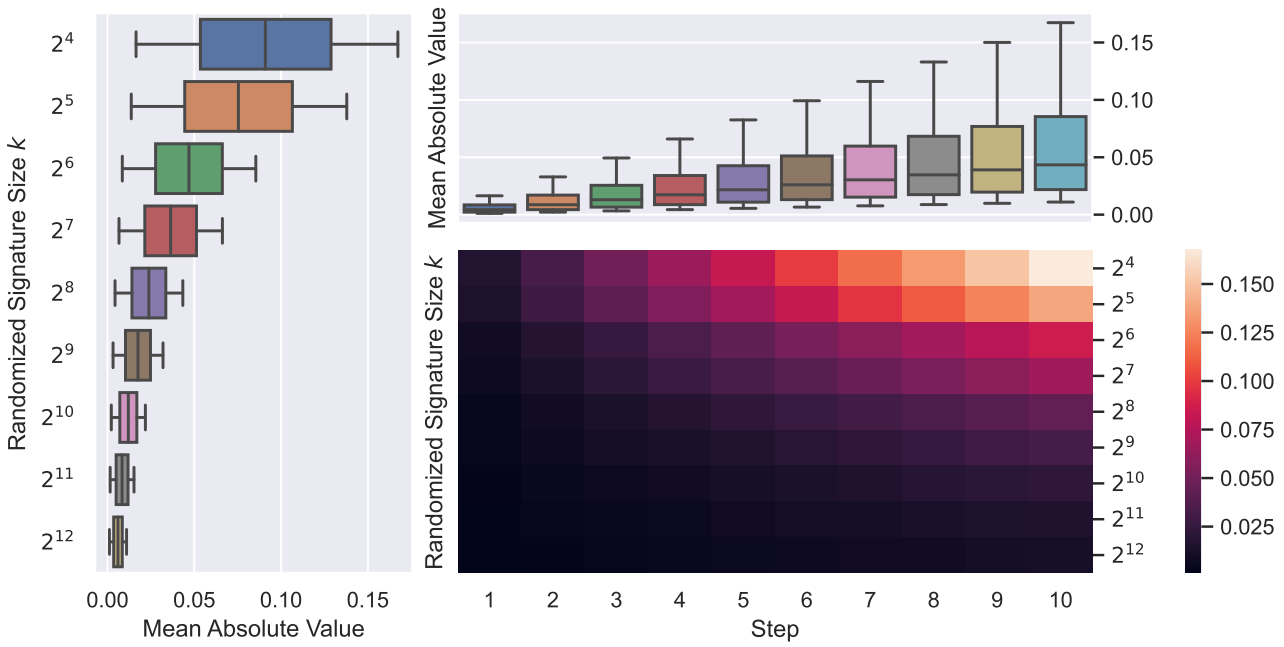

Figure 11. Ablation NONE (OURS). Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Our modified version does not suffer from exploding signals.

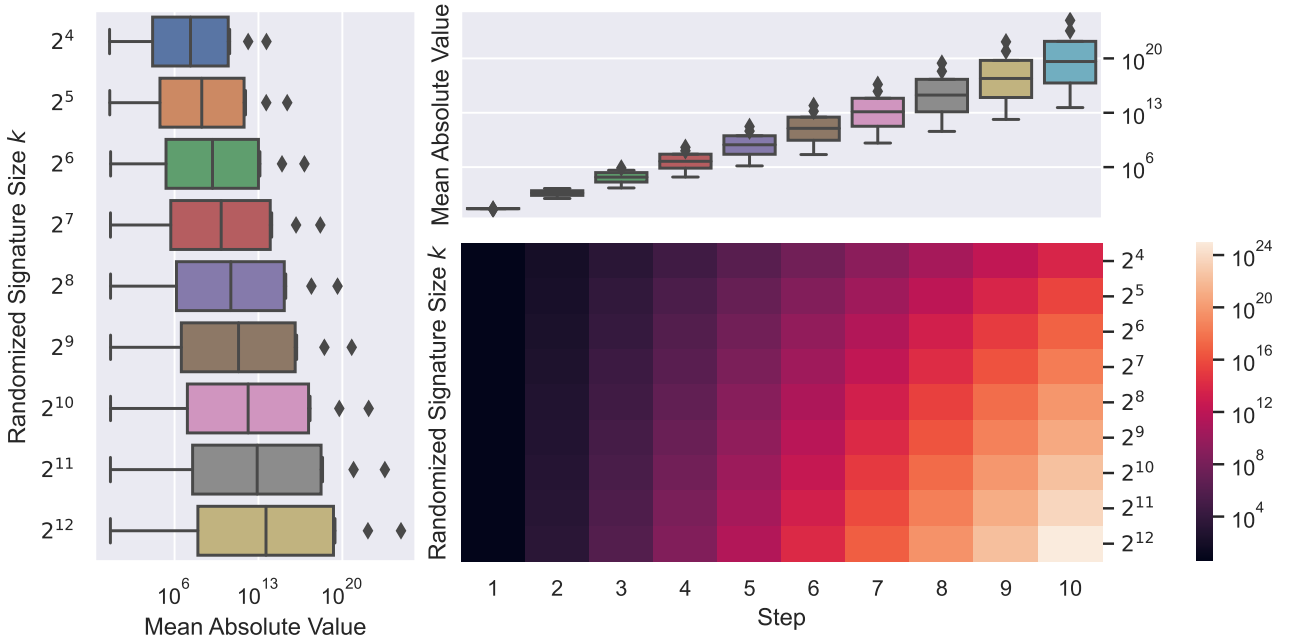

Figure 12. Ablation ALL (ORIGINAL). Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. The original computation of the randomized signature immediately produces exploding values.

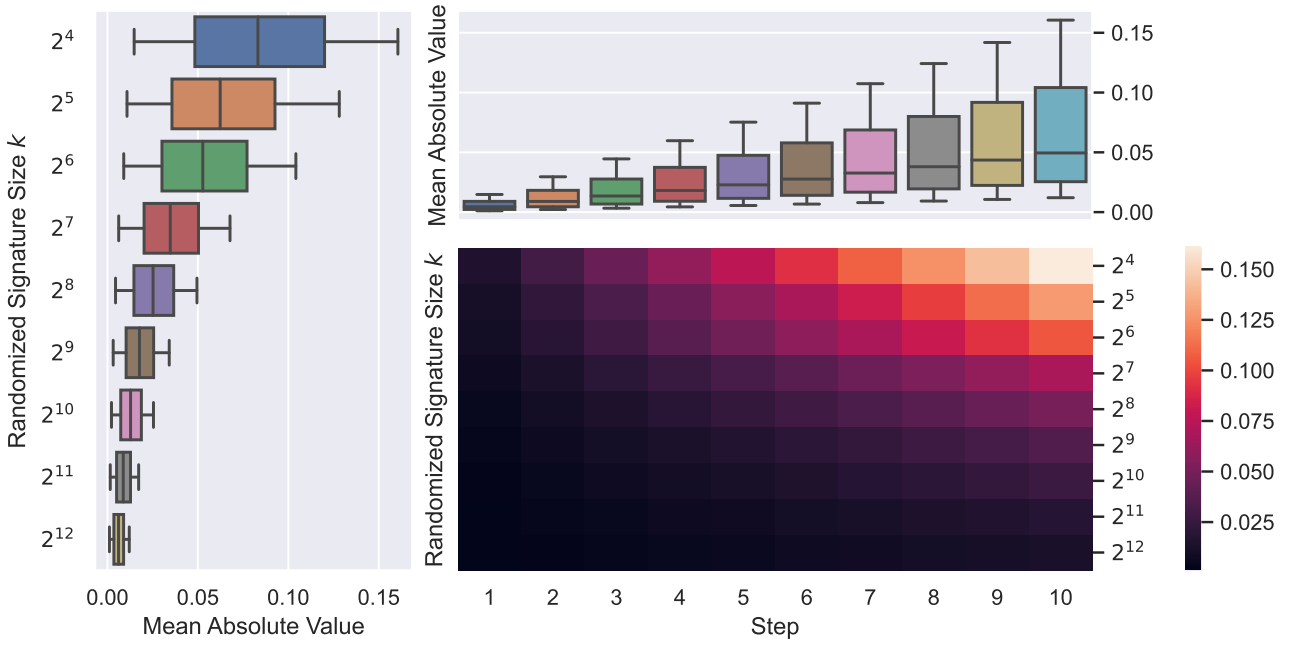

Figure 13. Ablation SPARSITY. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting the sparsity adjustment keeps computation stable at the expense of memory.

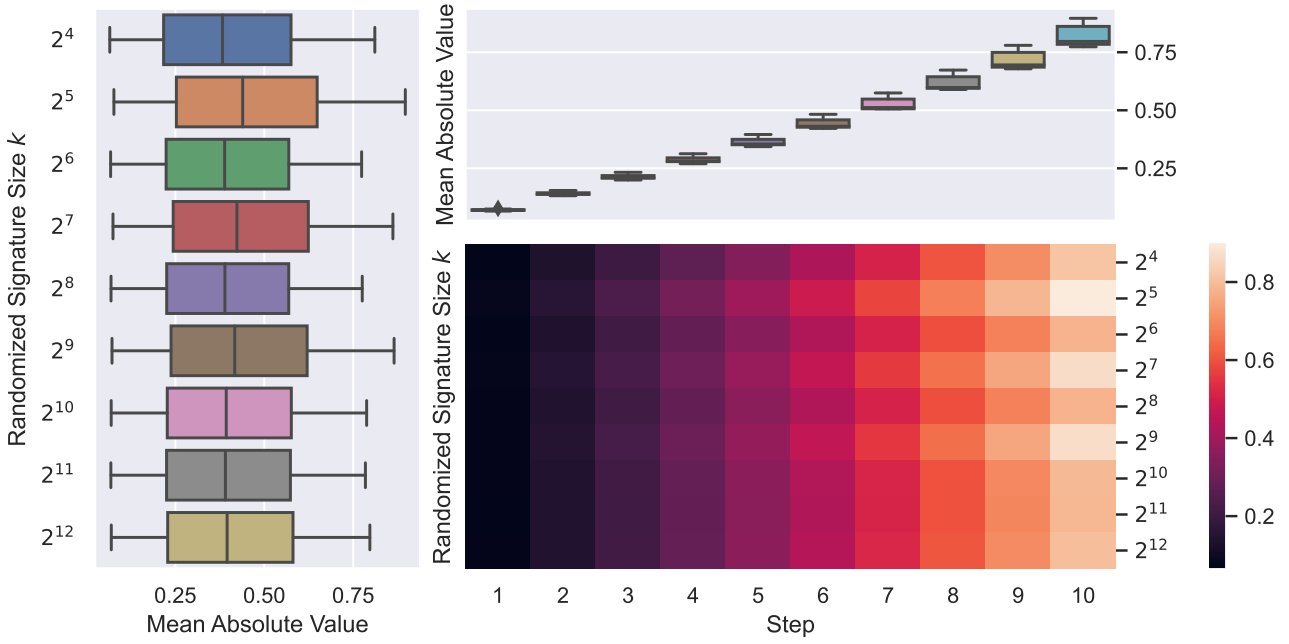

Figure 14. Ablation INITIALIZATION. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting the initialization adjustment results in a larger value magnitude.

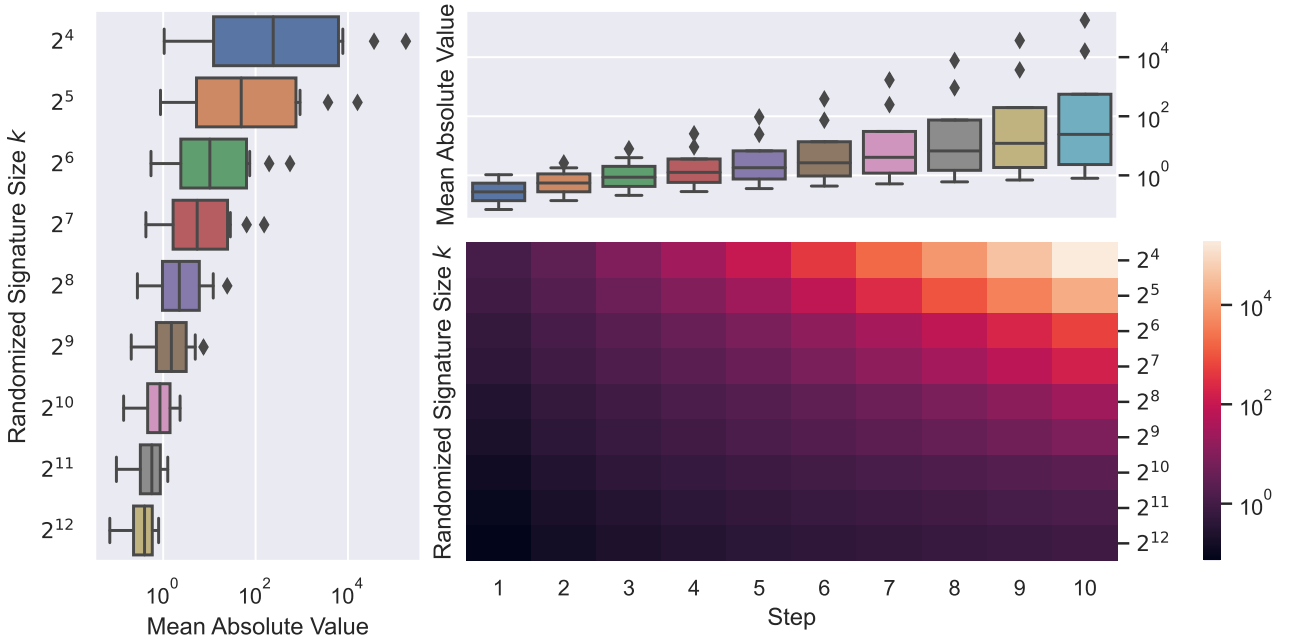

Figure 15. Ablation ACTIVATION. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting the activation adjustment renders the randomized signature computation susceptible to large signals.

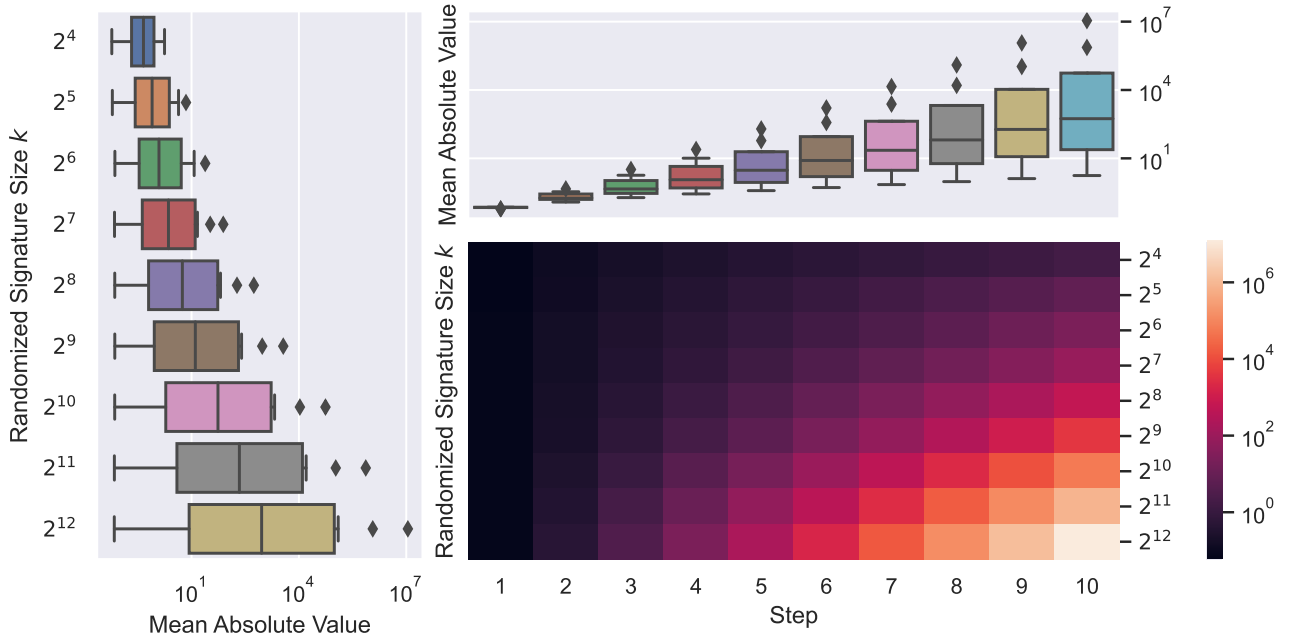

Figure 16. Ablation INITIALIZATION/SPARSITY. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting both initialization and sparsity adjustments results in exploding signals.

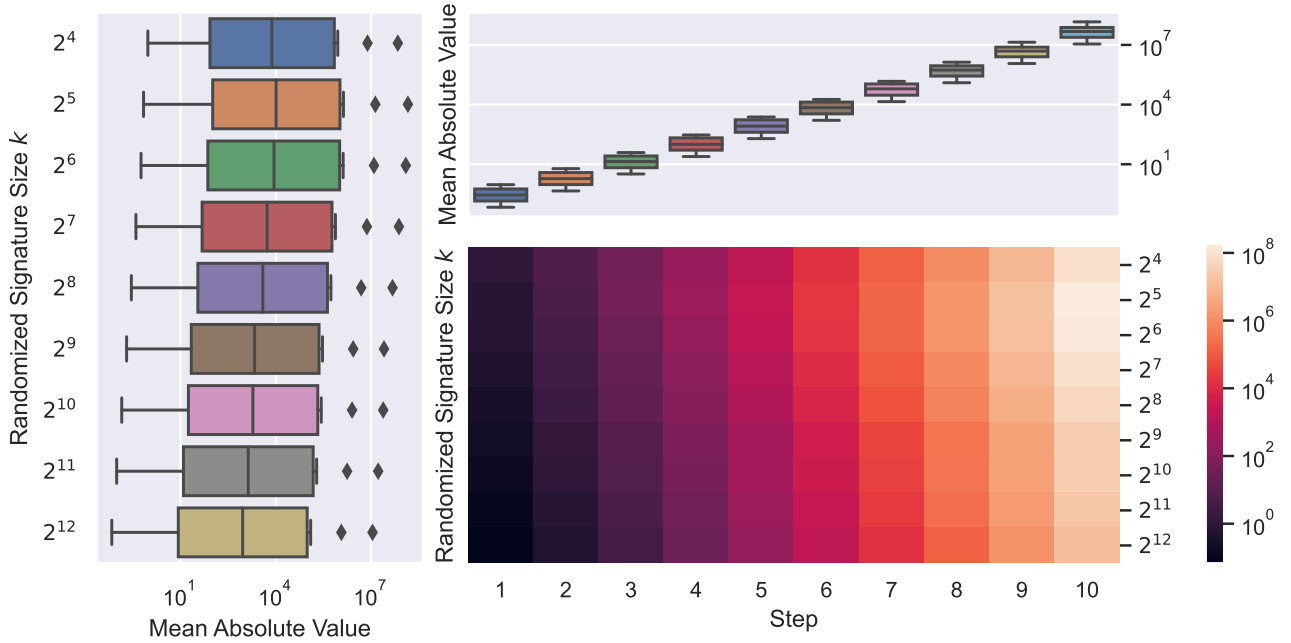

Figure 17. Ablation ACTIVATION/SPARSITY. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting both activation and sparsity adjustments leads to exploding values.

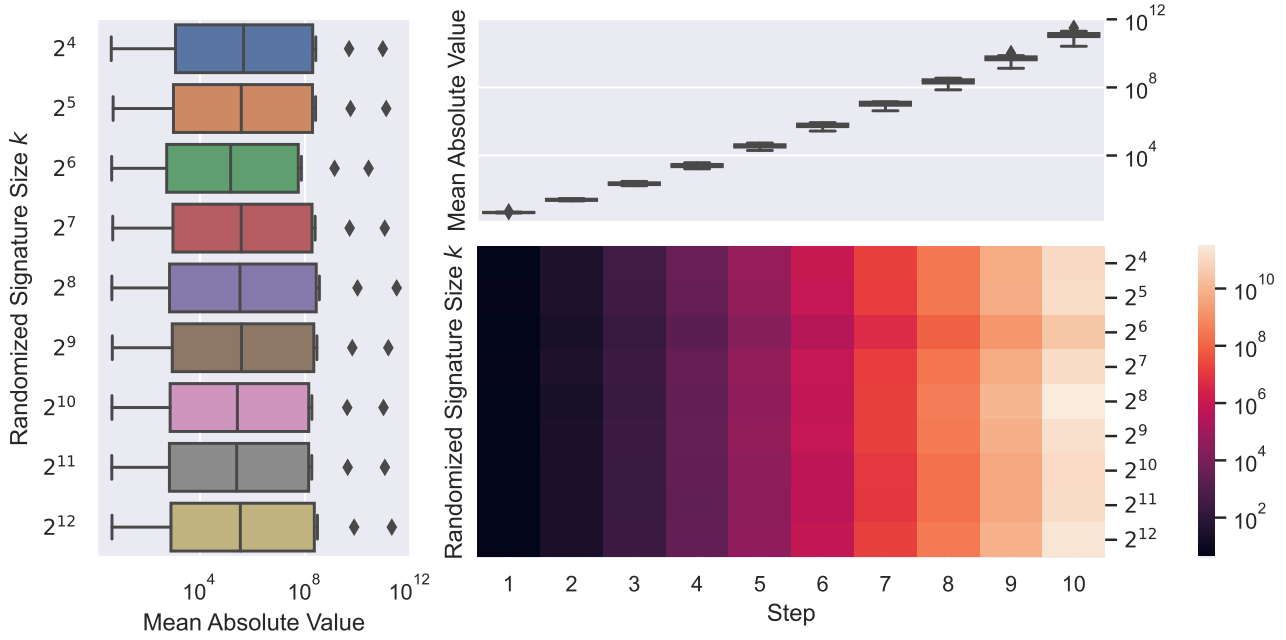

Figure 18. Ablation INITIALIZATION/ACTIVATION. Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Omitting both initialization and activation adjustments results in exploding signals.

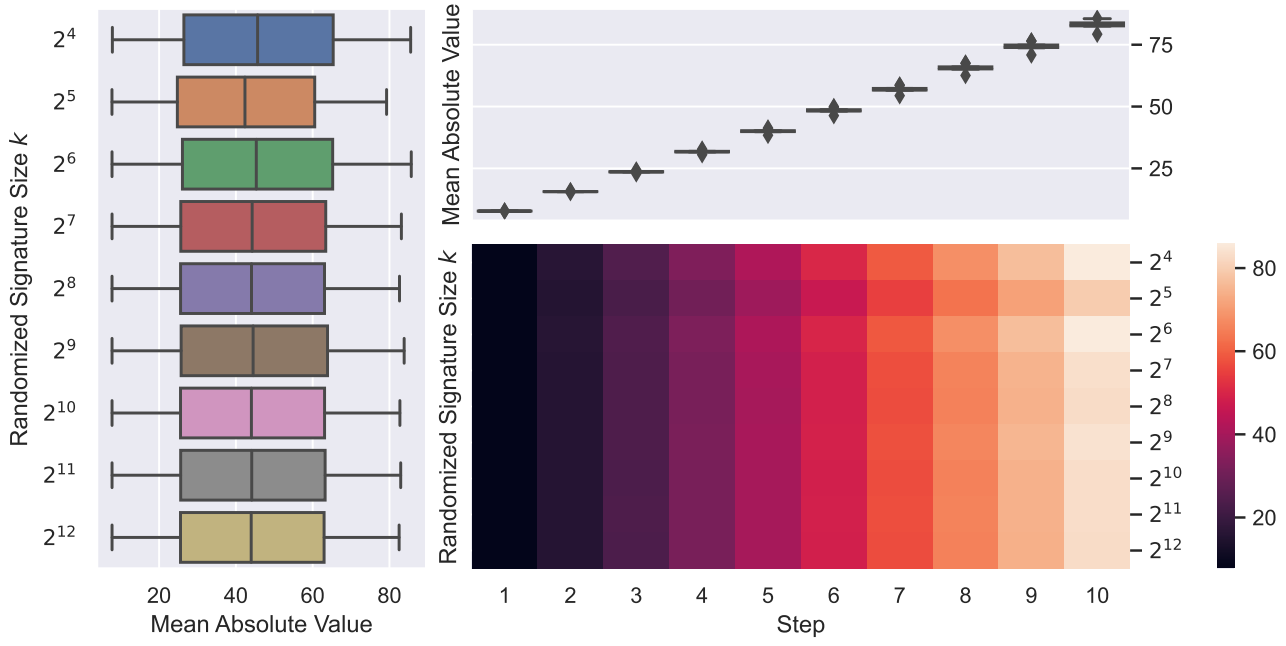

Figure 19. Ablation ACTIVATION (SIGMOID). Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Exchanging our activation adjustment with a $sigmoid$ non-linearity renders the randomized signature computation susceptible to large signals. Additionally, the consistency of the MAVs indicate a string saturating effect.

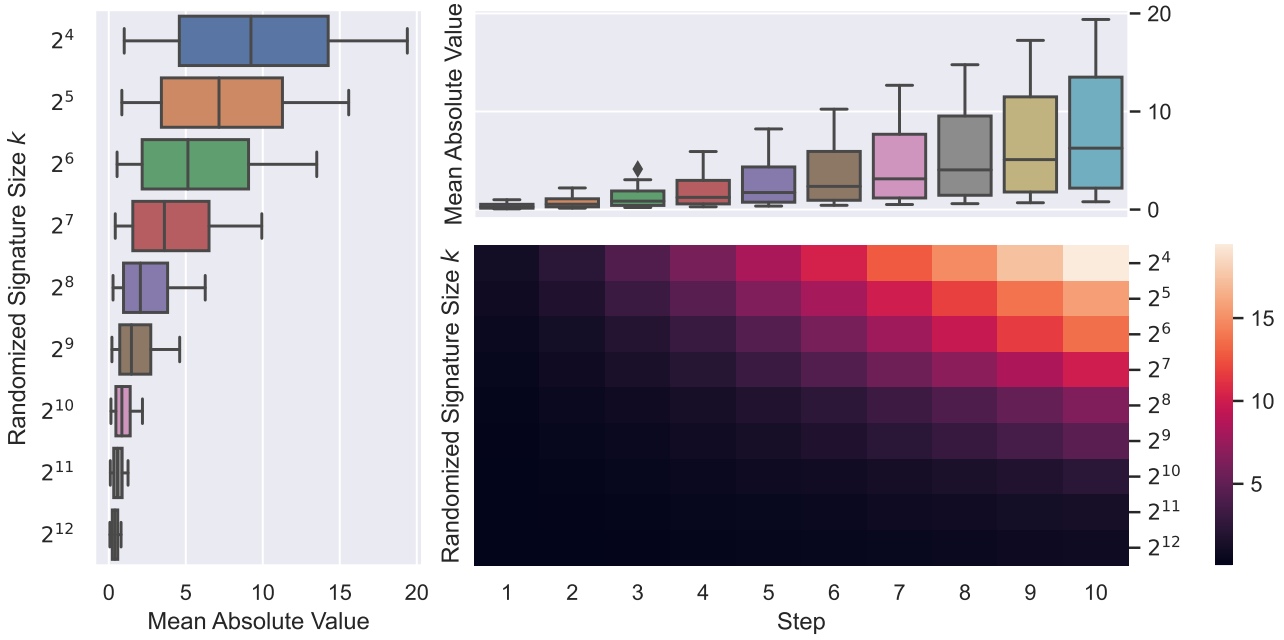

Figure 20. Ablation ACTIVATION (TANH). Lower right: Mean Absolute Values (MAVs) of randomized signature computation per step and randomized signature size. Upper right: Range of MAVs across randomized signature sizes per step. Left: Range of MAVs across steps per randomized signature size. Exchanging our activation adjustment with a $tanh$ non-linearity renders the randomized signature computation susceptible to large signals for small k. This effect is pronounced for higher step counts.

H. Hyperparameter Selection

For Gated GCNs and Graph Transformers we use hyperparameters that were already successfully applied to learn dependencies across longer node paths in (Dwivedi & Bresson, 2021; Dwivedi et al., 2022). We complement this by a manual hyperparameter search which results in the following hyperparameter search spaces.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	$\{1, \{2+2i\}_{i=0}^{5}, 16\}$	$\{6, 12, 32, 70, 100, 138\}$	$\{10^{-\{2,3,4\}}, 5 \times 10^{-4}\}$	-
GTs	$\{1, 2, 4, 8, 10\}$	$\{6, 12, 32, 80, 120\}$	$\{10^{-\{2,3,4\}}, 5 \times 10^{-4}\}$	$\{2^i\}_{i=0}^{4}$
G-Sigs.	$\{1, 2, 3\}$	$\{32+i\}_{i=0}^{64}$	$10^{[-3,-1]}$	$\{1, 2, 3\}$

Table 6. Hyperparameter settings for compared methods on CoMA.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	$\{1, 3, 4, 6, 8, 10, 12\}$	$\{6, 12, 32, 70, 100\}$	$\{10^{-\{2,3,4\}}, 5 \times 10^{-4}\}$	-
GTs	$\{1, 3, 4, 6, 8, 10\}$	$\{6, 12, 32, 80\}$	$\{10^{-\{2,3,4\}}, 5 \times 10^{-4}\}$	$\{1, 2, 4, 8\}$
G-Sigs.	$\{1, 2, 3\}$	$\{32+i\}_{i=0}^{64}$	$10^{[-3,-1]}$	$\{1, 2, 3\}$

Table 7. Hyperparameter settings for compared methods on ETA datasets.

Method	#Layers	Hidden Size	Learning Rate	#Heads	#Modes
ResNets	$\{4, 8, 16\}$	$\{32, 64, 128\}$	$10^{[-3,-1]}$	-	-
FNOs	$\{3, 4, 5\}$	$\{32, 128, 256\}$	$10^{[-3,-1]}$	-	$\{8, 12, 16, 32\}$
G-Sigs.	$\{1, 2, 3\}$	$\{32+i\}_{i=0}^{64}$	$10^{[-3,-1]}$	$\{1, 2, 3\}$	-

Table 8. Hyperparameter settings for compared methods on KPZ datasets.

All methods are trained for 1000 epochs with early stopping. The GNN methods are trained with either batch (Ioffe & Szegedy, 2015), or layer normalization (Ba et al., 2016), and by setting Dropout (Hinton et al., 2012) to 0.0 or 0.1. We test both averaging and summing over all nodes for the readout function of the respective GNNs. For all methods we use the LAMB (You et al., 2020) optimizer with an optional cosine annealed learning rate scheduling (Loshchilov & Hutter, 2017). The signature size hyperparameter of G-Signatures is selected from the set $\{8+i\}_{i=0}^{56}$. All G-Signatures hyperparameter searches are done in a non-exhaustive way.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	4	70	1.00×10^{-3}	-
GTs	10	80	1.00×10^{-3}	8
G.-Sigs.	1	73	3.30×10^{-3}	2

Table 9. Final hyperparameter settings for compared methods on CoMA. For G-Signatures, the final signature size is set to 63, and the final weight decay is set to 1.68×10^{-5}.

Method	#Layers	Hidden Size	Learning Rate	#Heads	#Modes
ResNets	8	128	1.00×10^{-2}	-	-
FNOs	4	128	1.00×10^{-2}	-	12
G.-Sigs.	1	84	1.05×10^{-3}	2	-

Table 10. Final hyperparameter settings for compared methods on the KPZ dataset with high noise. For G-Signatures, the final signature size is set to 27, and the final weight decay is set to 1.31×10^{-6}.

Method	#Layers	Hidden Size	Learning Rate	#Heads	#Modes
ResNets	8	128	1.00×10^{-2}	-	-
FNOs	4	128	1.00×10^{-2}	-	12
G.-Sigs.	1	89	1.19×10^{-3}	3	-

Table 11. Final hyperparameter settings for compared methods on the KPZ dataset with low noise. For G-Signatures, the final signature size is set to 47, and the final weight decay is set to 2.47×10^{-4}.

Method	#Layers	Hidden Size	Learning Rate	#Heads	#Modes
ResNets	8	128	1.00×10^{-2}	-	-
FNOs	4	128	1.00×10^{-2}	-	12
G.-Sigs.	1	81	1.01×10^{-3}	1	-

Table 12. Final hyperparameter settings for compared methods on the KPZ dataset with no noise. For G-Signatures, the final signature size is set to 61, and the final weight decay is set to 8.46×10^{-4}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	8	70	1.00×10^{-3}	-
GTs	6	32	1.00×10^{-3}	8
G.-Sigs.	1	86	5.87×10^{-3}	3

Table 13. Final hyperparameter settings for compared methods on the ETA dataset with 500 nodes and a sparsity of 0.0. For G-Signatures, the final signature size is set to 35, and the final weight decay is set to 2.55×10^{-4}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	70	1.00×10^{-3}	-
GTs	8	32	1.00×10^{-3}	8
G.-Sigs. 1	95	3.32×10^{-3}	3	

Table 14. Final hyperparameter settings for compared methods on the ETA dataset with 500 nodes and a sparsity of 0.5. For G-Signatures, the final signature size is set to 43, and the final weight decay is set to 1.82×10^{-6}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	70	1.00×10^{-3}	-
GTs	8	32	1.00×10^{-3}	8
G.-Sigs.	1	32	2.84×10^{-3}	2

Table 15. Final hyperparameter settings for compared methods on the ETA dataset with 500 nodes and a sparsity of 0.9. For G-Signatures, the final signature size is set to 27, and the final weight decay is set to 3.39×10^{-5}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	8	70	1.00×10^{-3}	-
GTs	6	32	1.00×10^{-3}	8
G.-Sigs.	1	79	2.91×10^{-3}	3

Table 16. Final hyperparameter settings for compared methods on the ETA dataset with 1000 nodes and a sparsity of 0.0. For G-Signatures, the final signature size is set to 40, and the final weight decay is set to 1.82×10^{-6}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	70	1.00×10^{-3}	-
GTs	8	32	1.00×10^{-3}	8
G.-Sigs.	1	71	3.51×10^{-3}	3

Table 17. Final hyperparameter settings for compared methods on the ETA dataset with 1000 nodes and a sparsity of 0.5. For G-Signatures, the final signature size is set to 26, and the final weight decay is set to 4.15×10^{-5}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	70	1.00×10^{-3}	-
GTs	8	32	1.00×10^{-3}	8
G.-Sigs.	1	50	4.13×10^{-3}	2

Table 18. Final hyperparameter settings for compared methods on the ETA dataset with 1000 nodes and a sparsity of 0.9. For G-Signatures, the final signature size is set to 40, and the final weight decay is set to 1.75×10^{-4}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	8	100	1.00×10^{-3}	-
GTs GTs	6	80	1.00×10^{-3}	8
G.-Sigs.	1	75	4.81×10^{-3}	3

Table 19. Final hyperparameter settings for compared methods on the ETA dataset with 2000 nodes and a sparsity of 0.0. For G-Signatures, the final signature size is set to 64, and the final weight decay is set to 2.29×10^{-6}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	100	1.00×10^{-3}	-
GTs GTs	8	80	1.00×10^{-3}	8
G.-Sigs.	1	59	5.06×10^{-3}	2

Table 20. Final hyperparameter settings for compared methods on the ETA dataset with 2000 nodes and a sparsity of 0.5. For G-Signatures, the final signature size is set to 53, and the final weight decay is set to 1.95×10^{-6}.

Method	#Layers	Hidden Size	Learning Rate	#Heads
GGCNs	10	100	1.00×10^{-3}	-
GTs GTs	8	80	1.00×10^{-3}	8
G.-Sigs.	1	77	2.70×10^{-3}	2

Table 21. Final hyperparameter settings for compared methods on the ETA dataset with 2000 nodes and a sparsity of 0.9. For G-Signatures, the final signature size is set to 36, and the final weight decay is set to 2.05×10^{-6}.