# MULTIRESOLUTION MATRIX FACTORIZATION AND WAVELET NETWORKS ON GRAPHS

**Truong Son Hy & Risi Kondor**
Department of Computer Science
University of Chicago
Chicago, IL 60637, USA
{hytruongson,risi}@uchicago.edu

## ABSTRACT

Multiresolution Matrix Factorization (MMF) is unusual amongst fast matrix factorization algorithms in that it does not make a low rank assumption. This makes MMF especially well suited to modeling certain types of graphs with complex multiscale or hierarchical strucutre. While MMF promises to yield a useful wavelet basis, finding the factorization itself is hard, and existing greedy methods tend to be brittle. In this paper, we propose a "learnable" version of MMF that carfully optimizes the factorization with a combination of reinforcement learning and Stiefel manifold optimization through backpropagating errors. We show that the resulting wavelet basis far outperforms prior MMF algorithms and provides the first version of this type of factorization that can be robustly deployed on standard learning tasks. Furthermore, we construct the wavelet neural networks (WNNs) learning graphs on the spectral domain with the wavelet basis produced by our MMF learning algorithm. Our wavelet networks are competitive against other state-of-the-art methods in molecular graphs classification and node classification on citation graphs. Our complete paper with the Appendix and more experiments is publicly available at https://arxiv.org/pdf/2111.01940.pdf. We release our implementation at https://github.com/risilab/Learnable_MMF/.

## 1 INTRODUCTION

In certain machine learning problems large matrices have complex hierarchical structures that traditional linear algebra methods based on the low rank assumption struggle to capture. Multiresolution matrix factorization (MMF) is a relatively little used alternative paradigm that is designed to capture structure at multiple different scales. MMF has been found to be particularly effective at compressing the adjacency or Laplacian matrices of graphs with complicated structure, such as social networks Kondor et al. (2014).

MMF factorizations have a number of advantages, including the fact that they are easy to invert and have an interpretation as a form of wavelet analysis on the matrix and consequently on the underlying graph. The wavelets can be used e.g., for finding sparse approximations of graph signals. Finding the actual MMF factorization however is a hard optimization problem combining elements of continuous and combinatorial optimization. Most of the existing MMF algorithms just tackle this with a variety of greedy heuristics and are consequently brittle: the resulting factorizations typically have large variance and most of the time yield factorizations that are far from the optimal Teneva et al. (2016); Ithapu et al. (2017); Ding et al. (2017).

The present paper proposes an alternative paradigm to MMF optimization based on ideas from deep learning. Specifically, we employ an iterative approach to optimizing the factorization based on backpropagating the factorization error and a reinforcement learning strategy for solving the combinatorial part of the problem. While more expensive than the greedy approaches, we find that the resulting "learnable" MMF produces much better quality factorizations and a wavelet basis that is smoother and better reflects the structure of the underlying matrix or graph. Unsurprisingly, this also means that the factorization performs better in downstream tasks.

To apply our learnable MMF algorithm to standard benchmark tasks, we also propose a wavelet extension of the Spectral Graph Networks algorithm of Bruna et al. (2014) which we call the Wavelet Neural Network (WNN). Our experiments show that the combination of learnable MMF optimization with WNNs achieves state of the art results on several graph learning tasks. Beyond just benchmark performance, the greatly improved stability of MMF optimization process and the similarity of the hierarchical structure of the factorization to the architecture of deep neural networks opens up the possibility of MMF being tightly integrated with other learning algorithms in the future.

## 2 Related work

Compressing and estimating large matrices has been extensively studied from various directions, including Drineas et al. (2006), Halko et al. (2011), Williams & Seeger (2001) Kumar et al. (2012), Mahoney (2011), Jenatton et al. (2010). Many of these methods come with explicit guarantees but typically make the assumption that the matrix to be approximated is low rank.

MMF is more closely related to other works on constructing wavelet bases on discrete spaces, including wavelets defined based on diagonalizing the diffusion operator or the normalized graph Laplacian Coifman & Maggioni (2006) Hammond et al. (2011) and multiresolution on trees Gavish et al. (2010) Lee et al. (2008). MMF has been used for matrix compression Teneva et al. (2016), kernel approximation Ding et al. (2017) and inferring semantic relationships in medical imaging data Ithapu et al. (2017).

Graph neural networks (GNNs) utilizing the generalization of convolution concept to graphs have been popularly applied to many learning tasks such as estimating quantum chemical computation, and modeling physical systems, etc. Spectral methods such as Bruna et al. (2014) provide one way to define convolution on graphs via convolution theorem and graph Fourier transform (GFT). To address the high computational cost of GFT, Xu et al. (2019) proposed to use the diffusion wavelet bases as previously defined by Coifman & Maggioni (2006) instead for a faster transformation.

## 3 Background on Multiresolution Matrix Factorization

The *Multiresolution Matrix Factorization* (MMF) of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a factorization of the form

$$\boldsymbol{A} = \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1,$$

where the $\boldsymbol{H}$ and $\boldsymbol{U}_1, \dots, \boldsymbol{U}_L$ matrices conform to the following constraints: (i) Each $\boldsymbol{U}_\ell$ is an orthogonal matrix that is a $k$-point rotation for some small $k$, meaning that it only rotates $k$ coordinates at a time; (ii) There is a nested sequence of sets $\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ such that the coordinates rotated by $\boldsymbol{U}_\ell$ are a subset of $\mathbb{S}_\ell$; and (iii) $\boldsymbol{H}$ is an $\mathbb{S}_L$-core-diagonal matrix that is diagonal with a an additional small $\mathbb{S}_L \times \mathbb{S}_L$ dimensional "core". Finding the best MMF factorization to a symmetric matrix $\boldsymbol{A}$ involves solving

$$\min_{\substack{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{H} \in \mathbb{H}_n^{\mathbb{S}_L}; \boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}}} ||\boldsymbol{A} - \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_1||. \tag{1}$$

Assuming that we measure error in the Frobenius norm, (1) is equivalent to

$$\min_{\substack{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}}} ||\boldsymbol{U}_L \dots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T||^2_{\text{resi}}, \tag{2}$$

where $||\cdot||^2_{\text{resi}}$ is the squared residual norm $||\boldsymbol{H}||^2_{\text{resi}} = \sum_{i \neq j; (i,j) \notin \mathbb{S}_L \times \mathbb{S}_L} |\boldsymbol{H}_{i,j}|^2$. There are two fundamental difficulties in MMF optimization: finding the optimal nested sequence of $\mathbb{S}_\ell$ is a combinatorially hard (e.g., there are $\binom{d_\ell}{k}$ ways to choose $k$ indices out of $\mathbb{S}_\ell$); and the solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$. The existing literature on solving this optimization problem Kondor et al. (2014) Teneva et al. (2016) Ithapu et al. (2017) Ding et al. (2017) has various heuristic elements and has a number of limitations. First of all, there is no guarantee that the greedy heuristics (e.g., clustering) used in selecting $k$ rows/columns $\mathbb{I}_\ell = \{i_1, .., i_k\} \subset \mathbb{S}_\ell$ for each rotation return a globally optimal factorization. Instead of direct optimization for each rotation $\boldsymbol{U}_\ell \triangleq \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell$ where $\boldsymbol{O}_\ell \in \mathbb{SO}(k)$ globally and simultaneously with the objective (1), Jacobi

MMFs (see Proposition 2 of Kondor et al. (2014)) apply the greedy strategy of optimizing them locally and sequentially. Again, this does not necessarily lead to a *globally* optimal combination of rotations. Furthermore, most MMF algorithms are limited to the simplest case of $k = 2$ where $\boldsymbol{U}_\ell$ is just a Givens rotation, which can be parameterized by a single variable, the rotation angle $\theta_\ell$. This makes it possible to optimize the greedy objective by simple gradient descent, but larger rotations would yield more expressive factorizations and better approximations.

In contrast, we propose an iterative algorithm to directly optimize the global MMF objective (1):

- We use gradient descent algorithm on the Stiefel manifold to optimize all rotations $\{\boldsymbol{U}_\ell\}_{\ell=1}^L$ *simultaneously*, whilst satisfying the orthogonality constraints. Importantly, the Stiefel manifold optimization is not limited to $k = 2$ case (Section 4).
- We formulate the problem of finding the optimal nested sequence $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ as learning a Markov Decision Process (MDP) that can be subsequently solved by the gradient policy method of Reinforcement Learning (RL), in which the RL agent (or stochastic policy) is modeled by graph neural networks (GNN) (Section 5).

We show that the resulting learning-based MMF algorithm outperforms existing greedy MMFs and other traditional baselines for matrix approximation in various scenarios (see Section 7).

## 4 Stiefel Manifold Optimization

The MMF optimization problem in (1) and (2) is equivalent to

$$\min_{\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]} \min_{\boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}} ||\boldsymbol{U}_L \dots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T||^2_{\text{resi}}, \tag{3}$$

In order to solve the inner optimization problem of (3), we consider the following generic optimization with orthogonality constraints:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times p}} \mathcal{F}(\boldsymbol{X}), \ \text{ s.t. } \ \boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{I}_p, \tag{4}$$

where $\boldsymbol{I}_p$ is the identity matrix and $\mathcal{F}(\boldsymbol{X}) : \mathbb{R}^{n \times p} \to \mathbb{R}$ is a differentiable function. The feasible set $\mathcal{V}_p(\mathbb{R}^n) = \{\boldsymbol{X} \in \mathbb{R}^{n \times p} : \boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{I}_p\}$ is referred to as the Stiefel manifold of $p$ orthonormal vectors in $\mathbb{R}^n$ that has dimension equal to $np - \frac{1}{2}p(p+1)$. We will view $\mathcal{V}_p(\mathbb{R}^n)$ as an embedded submanifold of $\mathbb{R}^{n \times p}$.

When there is more than one orthogonal constraint, (4) is written as

$$\min_{\boldsymbol{X}_1 \in \mathcal{V}_{p_1}(\mathbb{R}^{n_1}), \dots, \boldsymbol{X}_q \in \mathcal{V}_{p_q}(\mathbb{R}^{n_q})} \mathcal{F}(\boldsymbol{X}_1, \dots, \boldsymbol{X}_q) \tag{5}$$

where there are $q$ variables with corresponding $q$ orthogonal constraints. For example, in the MMF optimization problem (1), suppose we are already given $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ meaning that the indices of active rows/columns at each resolution were already determined, for simplicity. In this case, we have $q = L$ number of variables such that each variable $\boldsymbol{X}_\ell = \boldsymbol{O}_\ell \in \mathbb{R}^{k \times k}$, where $\boldsymbol{U}_\ell = \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell \in \mathbb{R}^{n \times n}$ in which $\mathbb{I}_\ell$ is a subset of $k$ indices from $\mathbb{S}_\ell$, must satisfy the orthogonality constraint. The corresponding objective function is

$$\mathcal{F}(\boldsymbol{O}_1, \dots, \boldsymbol{O}_L) = ||\boldsymbol{U}_L \dots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T||^2_{\text{resi}}. \tag{6}$$

Details about Stiefel manifold optimization is included in the Appendix.

## 5 Reinforcement Learning

### 5.1 Problem formulation

We formulate the problem of finding the optimal nested sequence of sets $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ as learning an RL agent in order to solve the MMF optimization in (1). There are two fundamental parts to index selection for each resolution level $\ell \in \{1, .., L\}$:

- Select $k$ indices $\mathbb{I}_\ell = \{i_1, .., i_k\} \subset \mathbb{S}_{\ell-1}$ to construct the corresponding rotation matrix $\boldsymbol{U}_\ell$ (see Section 4).
- Select the set of indices $\mathbb{T}_\ell \subset \mathbb{S}_{\ell-1}$ of rows/columns that are to be wavelets at this level, and then be eliminated by setting $\mathbb{S}_\ell = \mathbb{S}_{\ell-1} \setminus \mathbb{T}_\ell$. To reduce the computational cost, we assume that each resolution level has only one row/column to be selected as the wavelet (e.g., a single wavelet) such that $|\mathbb{T}_\ell| = 1$. That means the cardinality of $\mathbb{S}_\ell$ reduces by 1 after each level, $d_\ell = n - \ell$, and size of the core block of $\boldsymbol{H}$ is $(n - L) \times (n - L)$ that corresponds to exactly $n - L$ active rows/columns at the end.

## 5.2 Markov Decision Process

A key task for building our model is to specify our index selection procedure. We design an iterative index selection process and formulate it as a general decision process $M = (S, A, P, R, \gamma)$ as follows.

$S$ is the set of states (or state space) that consists of all possible intermediate and final states in which each state $s \in S$ is a tuple of $(\overline{\boldsymbol{A}}, \mathbb{S}, \ell)$ where $\ell$ indicates the resolution level, $\mathbb{S}$ indicates the set of active row/column indices, and $\overline{\boldsymbol{A}} = \boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ indicates the sub-matrix of $\boldsymbol{A}$ with indices of rows and columns are from $\mathbb{S}$ (e.g., $|\mathbb{S}| = d_\ell$, $\overline{\boldsymbol{A}} \in \mathbb{R}^{d_\ell \times d_\ell}$). We start at state $s_0 = (\boldsymbol{A}, [n], 0)$ where $\boldsymbol{A}$ is the input matrix that MMF tries to factorize (e.g., no rows/columns removed yet) and $[n]$ indicates all rows/columns are still active. The set of terminal (final) states $S^* \subset S$ includes every state $s^*$ that has $\ell = L$.

$A$ is the set of actions that describe the modification made to current state at each time step. An action $a \in A$ validly applied to a non-terminal state $s = (\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell)$ is a tuple $(\mathbb{I}, \mathbb{T})$ where $\mathbb{I} = \{i_1, .., i_k\} \subset \mathbb{S}$ is the set of $k$ indices corresponding to the rotation matrix $\boldsymbol{U}_{\ell+1}$, and $\mathbb{T} \subset \mathbb{S}$ is the set of wavelet indices to be returned at this level. This action transforms the state into the next one $s' = (\boldsymbol{A}_{\mathbb{S}',\mathbb{S}'}, \mathbb{S}', \ell + 1)$ where $\mathbb{S}' = \mathbb{S} \setminus \mathbb{T}$ meaning the set of active indices is further shrunk. The action is called invalid for the current state if and only if $\mathbb{I} \notin \mathbb{S}$ or $\mathbb{T} \notin \mathbb{S}$.

$P$ is the transition dynamics that specifies the possible outcomes of carrying out an action at time $t$ in which $p(a_\ell | s_\ell, .., s_0)$ is represented as a parameterized policy network $\pi_\theta$ with learnable parameters $\theta$. The whole trajectory is always started by the same $s_0$ and finished by a terminal state after exactly $L$ transitions:

$$s_0 \xrightarrow{a \sim \pi_\theta(\cdot|s_0)} s_1 \xrightarrow{a \sim \pi_\theta(\cdot|s_1)} s_2 \ldots s_{L-1} \xrightarrow{a \sim \pi_\theta(\cdot|s_{L-1})} s_L \in S^*,$$

that constructs the nested sequence.

$R(s^*)$ is the reward function that specifies the reward after reaching a terminal state $s^*$. The reward function is defined as negative of the MMF reconstruction loss such that

$$R(s^*) = -||\boldsymbol{A} - \boldsymbol{U}_1^T \ldots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \ldots \boldsymbol{U}_1||_F. \tag{7}$$

We want to maximize this final reward that is equivalent to minimize error of MMF in Frobenius norm (as in problem (1)). Evaluation of the reward requires the Stiefel manifold optimization (see Section 4) for rotations $\{\boldsymbol{U}_\ell\}_{\ell=1}^L$. Obviously, the final reward in Eq. (7) is the most important. However, to improve the training quality of the policy, we can define the intermediate reward $R(s)$ for non-terminal states $s = (\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell) \notin S^*$ as the immediate improvement of the $\ell$-th resolution ($L > \ell > 0$):

$$R(s) = -||[\boldsymbol{U}_\ell \boldsymbol{A}_{\ell-1} \boldsymbol{U}_\ell^T]_{\mathbb{S},\mathbb{S}}||^2_{\text{resi}}. \tag{8}$$

Along the trajectory $(s_0, s_1, .., s_L)$, we generate the corresponding sequence of rewards $(r_1, r_2, .., r_L)$ based on (7, 8).

$\gamma$ is the discount factor, a penalty to uncertainty of future rewards, $0 < \gamma \leq 1$. We define the return or discounted future reward $g_\ell$ for $\ell = 0, .., L - 1$ as

$$g_\ell = \sum_{k=0}^{L-\ell-1} \gamma^k r_{\ell+k+1}, \tag{9}$$

which in the case of $\gamma = 1$ indicates simply accumulating all the immediate rewards and the final reward along the trajectory.

## 5.3 Graph convolutional policy network

In this section, we design our policy network $\pi_\theta$ as a graph neural network (GNN) with the message passing scheme. We consider the symmetric matrix $\boldsymbol{A}$ being represented by a weighted undirected graph $\mathcal{G} = (V, E)$ in which $V$ is the set of nodes such that each node corresponds to a row/column of $\boldsymbol{A}$, and $E$ is the set of edges such that the edge $(i,j) \in E$ has the weight $\boldsymbol{A}_{i,j}$. As defined in Section 5.2, a state $s \in S$ is a tuple $(\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell)$ in which $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ is the sub-matrix restricted to the active rows/columns $\mathbb{S}$, and an action $a \in A$ is a tuple $(\mathbb{I}, \mathbb{T})$ in which $\mathbb{I}$ is the set of $k$ indices corresponding to the $(\ell+1)$-th rotation and $\mathbb{T}$ is the set of indices to be returned as wavelets. Practically, a state can be simply represented by a single binary vector such that if a bit is 1 then the corresponding index is active, without the need to explicitly storage matrix $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ that can be efficiently constructed from $\boldsymbol{A}$ by any numerical toolkit. Our GNN policy network $\pi_\theta(a|s)$ learns to encode the underlying graph represented by $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ and returns a sample of valid action such that $\mathbb{T} \subset \mathbb{I} \subset \mathbb{S}$. In Section 5.1, we assume that $\mathbb{T}$ contains only a single index that we will call as the *pivot* $i^*$ (e.g., $\mathbb{T} = \{i^*\}$). Thus, the task of our GNN model is to learn to select the pivot $i^*$ first, and then select the rest $k-1$ indices of $\mathbb{I}$ that are highly correlated to the pivot.

The simplest implementation of GNNs is Message Passing Neural Networks (MPNNs) Gilmer et al. (2017). Suppose that the node embeddings (messages) $\boldsymbol{M}_0 \in \mathbb{R}^{N\times F}$ are initialized by the input node features where $N$ is the number of nodes and $F$ is the number of features for each node. Iteratively, the messages are propagated from each node to its neighborhood, and then transformed by a combination of linear transformations and non-linearities, e.g.,

$$\hat{\boldsymbol{M}}_t = \boldsymbol{A}\boldsymbol{M}_{t-1}, \quad \boldsymbol{M}_t = \sigma(\hat{\boldsymbol{M}}_t \boldsymbol{W}_{t-1}), \tag{10}$$

where $\boldsymbol{A}$ is the adjacency matrix; $\hat{\boldsymbol{M}}_t$ and $\boldsymbol{M}_t \in \mathbb{R}^{N\times D}$ are the aggregated messages (by summing over each node's neighborhood) and the output messages at the $t$'th iteration, respectively; $\sigma$ is a element-wise non-linearity function (e.g., sigmoid, ReLU, etc.); and $\boldsymbol{W}$s are learnable weight matrices such that $\boldsymbol{W}_0 \in \mathbb{R}^{F\times D}$ and $\boldsymbol{W}_t \in \mathbb{R}^{D\times D}$ for $t > 0$. Basically, the set of learnable parameters $\theta$ of our policy network $\pi$ includes all $\boldsymbol{W}$s. In some cases, a graph Laplacian $\boldsymbol{L}$ is used instead of the adjacency matrix $\boldsymbol{A}$ in model (10), for example, graph Laplacian $\boldsymbol{L} = \boldsymbol{D}^{-1}\boldsymbol{A}$ or its symmetric normalized version $\tilde{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. One way to incorporate the set of active rows/columns/nodes $\mathbb{S}$ into our GNN model is by initializing the input node feature with a binary label ($f = 1$) such that a node $v$ has label 1 if the $v$-th row/column is still active, otherwise 0. For a more efficient implementation, we can execute the message passing in the block $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ only. Supposing that the message passing scheme is executed for $T$ iterations, we concatenate messages from every iteration together into the final embedding:

$$\boldsymbol{M} = \bigoplus_{t=1}^{T} \boldsymbol{M}_t \in \mathbb{R}^{N\times DT}. \tag{11}$$

Model (10) produces the embedding for each node that allows us to define a sampling procedure to select the pivot $i^*$. Given the final embedding $\boldsymbol{M}$ from Eq. (11), we define the probability $\boldsymbol{P}_i$ that node $i \in \mathbb{S}$ is being selected as the pivot as:

$$P_i = \frac{\exp(\hat{P}_i)}{\sum_{j\in\mathbb{S}} \exp(\hat{P}_j)}, \quad \text{where} \quad \hat{P}_i = \sum_f \boldsymbol{M}_{i,f}.$$

In order to make the sampling procedure differentiable for backpropagation, we apply the Gumbel-max trick Gumbel (1954) Maddison et al. (2014) Jang et al. (2017) that provides a simple and efficient way to draw sample $i^*$ as follows:

$$i^* = \text{one-hot}\Big(\arg\max_{i\in\mathbb{S}} \big[G_i + \log P_i\big]\Big),$$

where $G_i$ are i.i.d samples drawn from $\text{Gumbel}(0,1)$. Technically, the sample is represented by a one-hot vector such that the $i^*$-th element is 1. Similarly, $\mathbb{T}$, $\mathbb{I}$ and $\mathbb{S}$ are represented by vectors in $\{0,1\}^N$ in which a 1-element indicates the existence of the corresponding index in the set. Furthermore, the set union and minus operations (e.g., $\mathbb{S} \setminus \mathbb{T}$) can be easily done by vector addition and subtraction, respectively.

Given the pivot $i^*$, we compute the similarity score between $i^*$ and other nodes $i \in \mathbb{S}$ as $C_i = \langle \boldsymbol{M}_{i^*,:}, \boldsymbol{M}_{i,:}\rangle$. Finally, we sample $k-1$ nodes with the highest similarity scores to $i$ sequentially

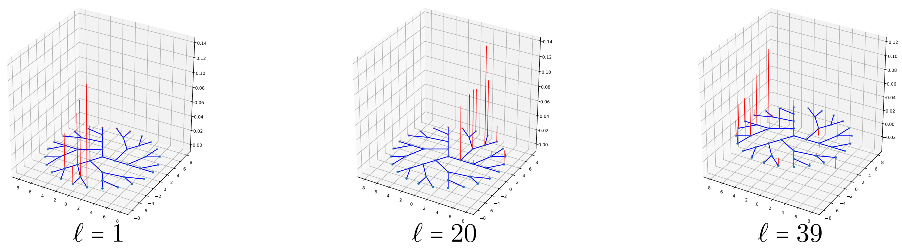

Figure 1: Visualization of some of the wavelets on the Cayley tree of 46 vertices. The low index wavelets (low $\ell$) are highly localized, whereas the high index ones are smoother and spread out over large parts of the graph.

(one-by-one) without replacement by the Gumbel-max trick, that completes our sampling procedure for action $a = (\mathbb{I}, \mathbb{T})$.

The REINFORCE Williams (1988) Williams (1992) Sutton et al. (2000) update rule for policy parameters is

$$\theta \leftarrow \theta + \eta\gamma^{\ell} g_{\ell} \nabla_{\theta} \log \pi_{\theta}(a_{\ell}|s_{\ell}) \quad \text{for} \quad \ell = 0, .., L-1; \tag{12}$$

where $\eta$ is the learning rate, that is used in training our policy network $\pi_\theta$ in Algorithm 1.

**Algorithm 1** MMF learning algorithm optimizing problem (1)

```
1: Input: Matrix $\boldsymbol{A}$, number of resolutions $L$, and constants $k$, $\gamma$, $\eta$, and $\omega$.
2: Initialize the policy parameter $\theta$ at random.
3: while true do
4:     Start from state $s_0$                                   ▷ $s_0 \triangleq (\boldsymbol{A}, [n], 0)$
5:     Initialize $\mathbb{S}_0 \leftarrow [n]$                 ▷ All rows/columns are active at the beginning.
6:     for $\ell = 0, .., L-1$ do
7:         Sample action $a_\ell = (\mathbb{I}_{\ell+1}, \mathbb{T}_{\ell+1})$ from $\pi_\theta(a_\ell|s_\ell)$.     ▷ See Section 5.3.
8:         $\mathbb{S}_{\ell+1} \leftarrow \mathbb{S}_\ell \setminus \mathbb{T}_{\ell+1}$     ▷ Eliminate the wavelet index (indices).
9:         $s_{\ell+1} \leftarrow (\boldsymbol{A}_{\mathbb{S}_{\ell+1}, \mathbb{S}_{\ell+1}}, \mathbb{S}_{\ell+1}, \ell+1)$     ▷ New state with a smaller active set.
10:    end for
11:    Given $\{\mathbb{I}_\ell\}_{\ell=1}^L$, minimize objective (6) by Stiefel manifold optimization to find $\{\boldsymbol{O}_\ell\}_{\ell=1}^L$.  ▷ $\boldsymbol{U}_\ell = \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell$
12:    for $\ell = 0, .., L-1$ do
13:        Estimate the return $g_\ell$ based on Eq. (7), Eq. (8), and Eq. (9).
14:        $\theta \leftarrow \theta + \eta\gamma^\ell g_\ell \nabla_\theta \log \pi_\theta(a_\ell|s_\ell)$     ▷ REINFORCE policy update in Eq. (12)
15:    end for
16:    Terminate if we already completed $\omega$ episodes.
17: end while
```

### 5.4 The learning algorithm

Putting everything together, our MMF learning algorithm is sketched in Algorithm 1. Iteratively: (1) we sample a trajectory by running the policy network that indicates the indices for rotation and wavelet for each resolution, (2) we apply the Stiefel manifold optimization to find the rotations, and (3) we compute the future rewards and update the parameters of the policy network by REINFORCE accordingly. The learning terminates when we complete $\omega$ iterations or episodes. In all our experiments, the learning algorithm can get to sufficient precision in matrix approximation after $\omega = 256$ episodes. Figure 1 depicts the wavelet bases at different levels of resolution.

## 6 Wavelet Networks on Graphs

### 6.1 Motivation

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a graph Fourier transform. Shuman et al. (2013) defines graph Fourier transform (GFT) on a graph $\mathcal{G} = (V, E)$ of a graph signal $\boldsymbol{f} \in \mathbb{R}^n$ (that is understood as a function $f : V \to \mathbb{R}$ defined on the vertices of the graph) as $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$, and the inverse graph Fourier transform as $\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$. Analogously to the classical Fourier transform, GFT provides a way to represent a graph signal in two domains: the vertex domain and the graph spectral domain; to filter graph signal according to smoothness; and to define the graph convolution operator, denoted as $*_{\mathcal{G}}$:

$$\boldsymbol{f} *_{\mathcal{G}} \boldsymbol{g} = \boldsymbol{U}\big((\boldsymbol{U}^T \boldsymbol{g}) \odot (\boldsymbol{U}^T \boldsymbol{f})\big), \tag{13}$$

where $\boldsymbol{g}$ denotes the convolution kernel, and $\odot$ is the element-wise Hadamard product. If we replace the vector $\boldsymbol{U}^T \boldsymbol{g}$ by a diagonal matrix $\tilde{\boldsymbol{g}}$, then we can rewrite the Hadamard product in Eq. (13) to matrix multiplication as $\boldsymbol{U}\tilde{\boldsymbol{g}}\boldsymbol{U}^T \boldsymbol{f}$ (that is understood as filtering the signal $\boldsymbol{f}$ by the filter $\tilde{\boldsymbol{g}}$). Based on GFT, Bruna et al. (2014) and Defferrard et al. (2016) construct convolutional neural networks (CNNs) learning on spectral domain for discrete structures such as graphs. However, there are two fundamental limitations of GFT:

- High computational cost: eigendecomposition of the graph Laplacian has complexity $O(n^3)$, and "Fourier transform" itself involves multiplying the signal with a dense matrix of eigenvectors.
- The graph convolution is not localized in the vertex domain, even if the graph itself has well defined local communities.

To address these limitations, we propose a modified spectral graph network based on the MMF wavelet basis rather than the eigenbasis of the Laplacian. This has the following advantages: (i) the wavelets are generally localized in both vertex domain and frequency, (ii) the individual basis transforms are sparse, and (iii) MMF provides a computationally efficient way of decomposing graph signals into components at different granularity levels and an excellent basis for sparse approximations.

### 6.2 Network construction

In the case $\boldsymbol{A}$ is the normalized graph Laplacian of a graph $\mathcal{G} = (V, E)$, the wavelet transform (up to level $L$) expresses a graph signal (function over the vertex domain) $f : V \to \mathbb{R}$, without loss of generality $f \in \mathbb{V}_0$, as:

$$f(v) = \sum_{\ell=1}^{L} \sum_{m} \alpha_m^\ell \psi_m^\ell(v) + \sum_{m} \beta_m \phi_m^L(v), \quad \text{for each } \ v \in V,$$

where $\alpha_m^\ell = \langle f, \psi_m^\ell \rangle$ and $\beta_m = \langle f, \phi_m^L \rangle$ are the wavelet coefficients. At each level, a set of coordinates $\mathbb{T}_\ell \subset \mathbb{S}_{\ell-1}$ are selected to be the wavelet indices, and then to be eliminated from the active set by setting $\mathbb{S}_\ell = \mathbb{S}_{\ell-1} \setminus \mathbb{T}_\ell$ (see Section 5.1). Practically, we make the assumption that we only select 1 wavelet index for each level (see Section 5.1) that results in a single mother wavelet $\psi^\ell = [\boldsymbol{A}_\ell]_{i^*,:}$ where $i^*$ is the selected index (see Section 5.3). We get exactly $L$ mother wavelets $\overline{\psi} = \{\psi^1, \psi^2, \dots, \psi^L\}$. On the another hand, the active rows of $\boldsymbol{H} = \boldsymbol{A}_L$ make exactly $N - L$ father wavelets $\overline{\phi} = \{\phi_m^L = \boldsymbol{H}_{m,:}\}_{m \in \mathbb{S}_L}$. In total, a graph of $N$ vertices has exactly $N$ wavelets (both mothers and fathers). Analogous to the convolution based on GFT Bruna et al. (2014), each convolution layer $k = 1, .., K$ of our wavelet network transforms an input vector $\boldsymbol{f}^{(k-1)}$ of size $|V| \times F_{k-1}$ into an output $\boldsymbol{f}^{(k)}$ of size $|V| \times F_k$ as

$$\boldsymbol{f}_{:,j}^{(k)} = \sigma\left(\boldsymbol{W} \sum_{i=1}^{F_{k-1}} \boldsymbol{g}_{i,j}^{(k)} \boldsymbol{W}^T \boldsymbol{f}_{:,i}^{(k-1)}\right) \quad \text{for } \ j = 1, \dots, F_k, \tag{14}$$

where $\boldsymbol{W}$ is our wavelet basis matrix as we concatenate $\overline{\phi}$ and $\overline{\psi}$ column-by-column, $\boldsymbol{g}_{i,j}^{(k)}$ is a parameter/filter in the form of a diagonal matrix learned in spectral domain, and $\sigma$ is an element-wise linearity (e.g., ReLU, sigmoid, etc.).

## 7 Molecular graphs classification

We trained and evaluated our wavelet networks (WNNs) on standard graph classification benchmarks including four bioinformatics datasets: (1) MUTAG, which is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds with 7 discrete labels Debnath et al. (1991); (2) PTC, which

Table 1: Molecular graphs classification. Baseline results are taken from Maron et al. (2019).

| **Method** | **MUTAG** | **PTC** | **PROTEINS** | **NCI1** |
|---|---|---|---|---|
| DGCNN Zhang et al. (2018) | 85.83 ± 1.7 | 58.59 ± 2.5 | 75.54 ± 0.9 | 74.44 ± 0.5 |
| PSCN Niepert et al. (2016) | 88.95 ± 4.4 | 62.29 ± 5.7 | 75 ± 2.5 | 76.34 ± 1.7 |
| DCNN Atwood & Towsley (2016) | N/A | N/A | 61.29 ± 1.6 | 56.61 ± 1.0 |
| CCN Hy et al. (2018) | **91.64** ± **7.2** | **70.62** ± **7.0** | N/A | 76.27 ± 4.1 |
| GK Shervashidze et al. (2009) | 81.39 ± 1.7 | 55.65 ± 0.5 | 71.39 ± 0.3 | 62.49 ± 0.3 |
| RW Vishwanathan et al. (2010) | 79.17 ± 2.1 | 55.91 ± 0.3 | 59.57 ± 0.1 | N/A |
| PK Neumann et al. (2016) | 76 ± 2.7 | 59.5 ± 2.4 | 73.68 ± 0.7 | 82.54 ± 0.5 |
| WL Shervashidze et al. (2011) | 84.11 ± 1.9 | 57.97 ± 2.5 | 74.68 ± 0.5 | **84.46** ± **0.5** |
| IEGN Maron et al. (2019) | 84.61 ± 10 | 59.47 ± 7.3 | 75.19 ± 4.3 | 73.71 ± 2.6 |
| **MMF** | 86.31 ± 9.47 | 67.99 ± 8.55 | **78.72** ± **2.53** | 71.04 ± 1.53 |

consists of 344 chemical compounds with 19 discrete labels that have been tested for positive or negative toxicity in lab rats Toivonen et al. (2003); (3) PROTEINS, which contains 1,113 molecular graphs with binary labels, where nodes are secondary structure elements (SSEs) and there is an edge between two nodes if they are neighbors in the amino-acid sequence or in 3D space Borgwardt et al. (2005); (4) NCI1, which has 4,110 compounds with binary labels, each screened for activity against small cell lung cancer and ovarian cancer lines Wale et al. (2008). Each molecule is represented by an adjacency matrix, and we represent each atomic type as a one-hot vector and use them as the node features.

We factorize all normalized graph Laplacian matrices in these datasets by MMF with $K = 2$ to obtain the wavelet bases. Again, MMF wavelets are **sparse** and suitable for fast transform via sparse matrix multiplication, with the following average percentages of non-zero elements for each dataset: $19.23\%$ (MUTAG), $18.18\%$ (PTC), $2.26\%$ (PROTEINS) and $11.43\%$ (NCI1).

Our WNNs contain 6 layers of spectral convolution, 32 hidden units for each node, and are trained with 256 epochs by Adam optimization with an initial learning rate of $10^{-3}$. We follow the evaluation protocol of 10-fold cross-validation from Zhang et al. (2018). We compare our results to several deep learning methods and popular graph kernel methods. Baseline results are taken from Maron et al. (2019). Our WNNs outperform 7/8, 7/8, 8/8, and 2/8 baseline methods on MUTAG, PTC, PROTEINS, and NCI1, respectively (see Table 1).

Experiments on node classification on citation graphs and matrix factorization are included in the full paper at `https://arxiv.org/pdf/2111.01940.pdf`.

## 8 Software

We implemented our learning algorithm for MMF and the wavelet networks by PyTorch deep learning framework (Paszke et al., 2019). We released our implementation at `https://github.com/risilab/Learnable_MMF/`.

## 9 Conclusions

In this paper we introduced a general algorithm based on reinforcement learning and Stiefel manifold optimization to optimize Multiresolution Matrix Factorization (MMF). We find that the resulting learnable MMF consistently outperforms the existing greedy and heuristic MMF algorithms in factorizing and approximating hierarchical matrices. Based on the wavelet basis returned from our learning algorithm, we define a corresponding notion of spectral convolution and construct a wavelet neural network for graph learning problems. Thanks to the sparsity of the MMF wavelets, the wavelet network can be efficiently implemented with sparse matrix multiplication. We find that this combination of learnable MMF factorization and spectral wavelet network yields state of the art results on standard node classification and molecular graph classification.

## References

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 2001–2009, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1: i47–56, 2005.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.04.004. Special Issue: Diffusion Maps and Wavelets.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991. doi: 10.1021/jm00106a046.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yi Ding, Risi Kondor, and Jonathan Eskreis-Winkler. Multiresolution kernel approximation for gaussian process regression. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Petros Drineas, R. Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36:158–183, 2006.

Matan Gavish, Boaz Nadler, and Ronald R. Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 367–374, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017.

E. J. Gumbel. Statistical theory of extreme values and some practical applications: a series of lectures. *US Govt. Print. Office*, Number 33, 1954.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. doi: 10.1137/090771806.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005.

Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M. Anderson, , and Risi Kondor. Predicting molecular properties with covariant compositional networks. *The Journal of Chemical Physics*, 148, 2018.

Vamsi K. Ithapu, Risi Kondor, Sterling C. Johnson, and Vikas Singh. The incremental multiresolution matrix factorization algorithm. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 692–701, 2017. doi: 10.1109/CVPR.2017.81.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 366–373, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Risi Kondor, Nedelina Teneva, and Vikas Garg. Multiresolution matrix factorization. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1620–1628, Bejing, China, 22–24 Jun 2014. PMLR.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nystrom method. *Journal of Machine Learning Research*, 13(34):981–1006, 2012.

Ann B. Lee, Boaz Nadler, and Larry Wasserman. Treelets–An adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, 2(2):435 – 471, 2008. doi: 10.1214/07-AOAS137.

Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3 (2):123–224, February 2011. ISSN 1935-8237. doi: 10.1561/2200000035.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019.

M. Neumann, R. Garnett, C. Baukhage, and K. Kersting. Propagation kernels: Efficient graph kernels from propagated information. *Machine Learning*, 102:209–245, 2 2016.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2014–2023, New York, New York, USA, 20–22 Jun 2016. PMLR.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 488–495, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561, 2011.

David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. doi: 10.1109/MSP.2012.2235192.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000.

Nedelina Teneva, Pramod Kaushik Mudrakarta, and Risi Kondor. Multiresolution matrix compression. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 1441–1449, Cadiz, Spain, 09–11 May 2016. PMLR.

Hannu Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the Predictive Toxicology Challenge 2000–2001. *Bioinformatics*, 19(10):1183–1193, 07 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg130.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010. ISSN 1532-4435.

Nikil Wale, Ian Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowl. Inf. Syst.*, 14:347–375, 03 2008. doi: 10.1109/ICDM.2006.39.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001.

Ronald J. Williams. Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, Northeastern University, College of Computer Science, 1988.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696.

Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.