scProfiterole: Clustering of Single-Cell Proteomic Data Using Graph Contrastive Learning via Spectral Filters

Mustafa Coşkun[1], Filipa Blasco Lopes[2,3], Pınar Kubilay Tolunay[4], Mark R. Chance[2,3], and Mehmet Koyutürk[2,5]

[1] Department of Artificial Intelligence and Data Engineering, Ankara University
[2] Center for Proteomics and Bioinformatics, Case Western Reserve University
[3] Department of Nutrition, Case Western Reserve University
[4] Department of Medical Oncology, Ankara University School of Medicine
[5] Department of Computer and Data Sciences, Case Western Reserve University
coskunmustafa@ankara.edu.tr, mehmet.koyuturk@case.edu

Abstract. Novel technologies for the acquisition of protein expression data at the single cell level are emerging rapidly. Although there exists a substantial body of computational algorithms and tools for the analysis of single cell gene expression (scRNAseq) data, tools for even basic tasks such as clustering or cell type identification for single cell proteomic (scProteomics) data are relatively scarce. Adoption of algorithms that have been developed for scRNAseq into scProteomics is challenged by the larger number of drop-outs, missing data, and noise in single cell proteomic data. Graph contrastive learning (GCL) on cell-to-cell similarity graphs derived from single cell protein expression profiles show promise in cell type identification. However, missing edges and noise in the cell-to-cell similarity graph requires careful design of convolution matrices to overcome the imperfections in these graphs. Here, we introduce scProfiterole (Single Cell Proteomics Clustering via Spectral Filters), a computational framework to facilitate effective use of spectral graph filters in GCL-based clustering of single cell proteomic data. Since clustering assumes a homophilic network topology, we consider three types of homophilic filters: (i) random walks, (ii) heat kernels, (iii) beta kernels. Since direct implementation of these filters is computationally prohibitive, the filters are either truncated or approximated in practice. To overcome this limitation, scProfiterole uses Arnoldi orthonormalization to implement polynomial interpolations of any given spectral graph filter. Our results on comprehensive single cell proteomic data show that (i) graph contrastive learning with learnable polynomial coefficients that are carefully initialized improves the effectiveness and robustness of cell type identification, (ii) heat kernels and beta kernels improve clustering performance over adjacency matrices or random walks, and (iii) polynomial interpolation of complex filters outperforms approximation or truncation. The source code for scProfiterole is available at https://github.com/mustafaCoskunAgu/scProfiterole

Keywords: Single cell proteomics, Clustering, Cell type identification, Graph contrastive learning, Spectral graph filters, Polynomial Interpolation

1 Introduction

Recent technological advances in single-cell RNA sequencing (scRNA-seq) have enabled monitoring of transcriptomic states at the resolution of individual cells [36]. Transcriptomic states provide useful information to characterize cellular heterogeneity [27], cell-to-cell interactions [26,40], and spatio-temporal dynamics of cell states [18]. However, mRNA expression exhibits limited correlation with protein abundance [30,31], thus provides limited information on the post-translational states of the cells [2]. Proteins serve as the primary actors of cellular function and signaling, acting through enzymatic activity, complex formation, and post-translational modifications. Due to the difficulties in the identification and quantification of peptide sequences and the transient nature of post-translational dynamics, single-cell proteomics has long been limited to monitoring the expression of a small set of selected proteins [1]. The emergence of recent technologies for proteome-wide single-cell proteomics (scProteomics) provides unprecedented opportunities to cell states by directly measuring functional cellular processes [14].

As single cell proteomic data emerges, the analysis of this novel type of omic data presents distinct computational challenges. Technical limitations in sample preparation, isotopic labeling, and mass spectrometry (MS) acquisition often introduce substantial peptide quantification uncertainty, missing data, and batch effects [34]. Recent studies demonstrate that graph-based deep learning, particularly graph contrastive learning (GCL), can denoise proteomic data and model uncertainty to a certain extent, enhancing the effectiveness of scProteomics in basic tasks such as clustering and cell type annotation [25]. Graph neural networks have also demonstrated success in analyzing scRNAseq data [4,39]. Yet, the application of graph representation learning to scProteomics faces two key challenges: (i) severe data sparsity and noise, which lead to unreliable cell-to-cell similarity graphs, and (ii) *over-smoothing* of signals with increasing number of layers in the neural network [5,24], which limit the application of GCLs to shallow networks on these datasets. Fig. 1 demonstrates the detrimental effect of oversmoothing on the clustering of scProteomics data via scPROTEIN [25], a GCL algorithm that uses Graph Convolutional Network (GCN) based encoders. This limitation restricts the ability to aggregate information from distant yet functionally related nodes—an essential aspect of modeling biological systems that often display long-range homophily [9]. Indeed, scPROTEIN uses a two-layer neural network, limiting the view of the GCN to shallow neighborhood aggregation.

We argue that spectral graph filters can overcome the limitations posed by the sparse and noisy nature of cell-to-cell similarity graphs in scProteomics without requiring a deeper neural network, by providing a broader perspective of graph topology. It has been repeatedly shown in the graph machine learning literature [20], as well as in other applications of graph machine learning in systems biology [19], that spectral graph filters are highly effective in overcoming imperfections in graph-structured data. Spectral graph filters transform graph topology by effectively operating in the eigenspace of the graph adjacency matrix or the graph Laplacian, where the filter specifies the part of the spectra to be emphasized. Since computing the eigendecomposition of the filter functions is cost-prohibitive, these filters are usually implemented using polynomial approximations, also enabling adaptive learning of the polynomial coefficients during training. An important design choice in this regard is the spectral filter that suits best to the application in hand, as well as the algorithms for efficient implementation of the filter. Since cell-to-cell similarity graphs are constructed to be homophilic by design, low-pass filters stand as the natural choice for cell type annotation.

Here, we present scProfiterole (Single Cell Proteomics Clustering via Spectral Filters), a framework that combines the interpretability of spectral graph theory with the power of graph contrastive learning. To provide a broad perspective on the use of low-pass filters in clustering single-cell proteomic data, scProfiterole implements three families of spectral filters: 1) Random-walk with restarts (RWR), 2) Heat kernels (HK), 3) Beta kernels (BK). These filters are visualized in Fig. 2. While RWRs are well-established and relatively straightforward to implement, the filters that

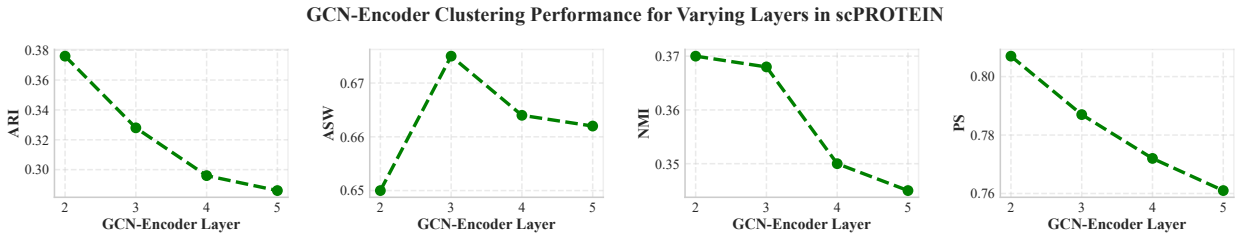

Fig. 1: **Performance of GCN encoders in clustering scProteomics data as a function of layer depth.** The performance of encoders using the adjacency matrix of the cell-to-cell similarity matrix for convolution on the `Scope2_Specht` dataset is shown. Over-smoothing leads to performance degradation with deeper architectures. ARI: Adjusted Rand Index, ASW: Average Silhouette Width, NMI: Normalized Mutual Information, PS: Purity Score.

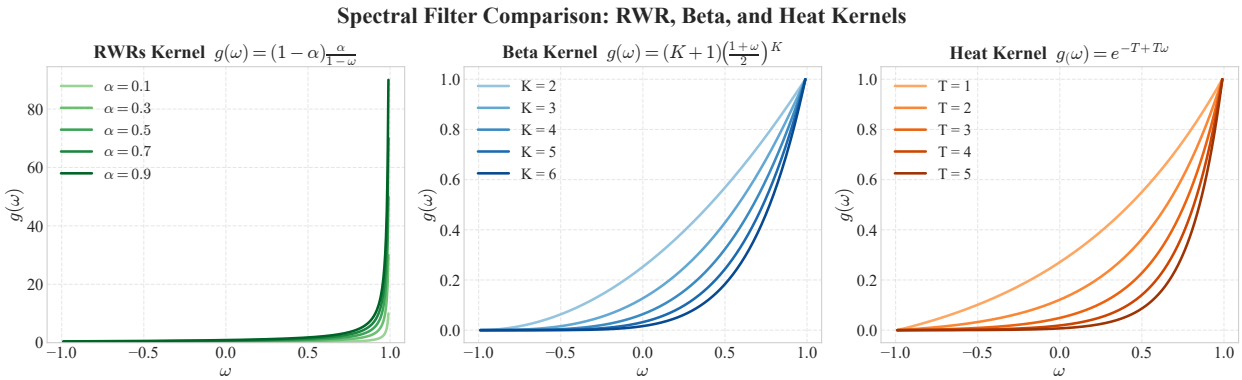

Fig. 2: **The landscape of spectral filters represented by the low-pass filter families implemented in** scProfiterole**.** The x-axis shows the range of the eigenvalues of the normalized adjacency matrix of a graph, the plots show the function represented by the specific type of filter with the specific value of the filter's parameter. The filters appear reversed (i.e, they are monotonically increasing functions of the eigenvalues) as compared to low-pass filters implemented on the graph Laplacian, since we express the filters in terms of the adjacency matrix. The filter represented by the graph adjacency matrix corresponds to the identity ($y = x$) line.

correspond to RWRs focus on a narrow band of the eigenspectra and the damping factor (α) has limited effect on the filter (Fig. 2, left). Heat kernels, also called continuous-time random walks [22], offer a more flexible and broader family of filters (Fig. 2, right), but they are more difficult to implement. Taylor approximation of Heat kernels is used for efficient implementation [22]. As an alternate strategy, we here use polynomial interpolations of the filter functions [7]. While the ill-conditioned nature of the linear system to be solved for polynomial interpolation poses numerical challenges, we overcome these challenges using Arnoldi orthonormalization [6]. Polynomial interpolation improves the fidelity of the implemented polynomial filters to the filter function. Finally, Beta kernels (Fig. 2, center) offer a family of filters that are polynomial by design, thus they can be implemented directly.

Our results on state-of-the-art scProteomic datasets demonstrate that scProfiterole enhances clustering performance and cell type annotation, outperforming traditional GCN and contrastive learning baselines. We observe that the performance of the spectral GCL, as well as the values of the learnable polynomial coefficients depend strongly on the spectral filter that is used for the initialization of these parameters. Heat kernels clearly deliver the best performance in general, and polynomial interpolation via Arnoldi orthonormalization enhances the performance of Heat kernels over Taylor approximation. Altogether, scProfiterole provides a principled and scalable foundation for graph contrastive learning on noisy, high-dimensional single-cell proteomic data, spearheading the application of graph representative learning to extend the boundaries of next-generation proteomics.

2 Methods

2.1 Background and Problem Formulation

Len $\boldsymbol{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times d}$ denote the cell-to-protein abundance matrix, where $\boldsymbol{X}(i, j)$ represents the expression of protein j in cell i, N is the number of cells, and d is the number of proteins. As in scRNAseq analysis, the first step in the analysis of scProteomics data is to use $\boldsymbol{X}$ to construct a cell-to-cell similarity graph [25]. Here, since our focus is on the utilization of the cell-to-cell similarity graph for cell clustering and cell-type annotation, we assume that the cell-to-cell similarity graph is readily available as an undirected, unweighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.Here, $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ denotes the set of the cells and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(v_i, v_j) \in \mathcal{E}$ indicates that cells i and j are similar to each other in terms of their protein expression profiles.

Li et al's scPROTEIN framework [25] uses unsupervised deep graph contrastive learning (GCL) to learn cell embeddings from $\mathcal{G}$ and $\boldsymbol{X}$. It comprises four key modules: 1) Data augmentation, 2) GCN-based encoder, 3) Node-level contrastive learning, 4) Alternating topology–attribute denoising. While effective, the standard GCN encoder primarily captures local neighborhood information and struggles with deeper architectures. As demonstrated Fig. 1, addition of more than two neural network layers in the GCN encoder causes over-smoothing [5,24,37], leading to degradation of performance in downstream tasks. In addition, when the adjacency matrix is used for convolution, each k-hop neighborhood contributes equally to the propagation of features, making the framework vulnerable to the imperfections in the underlying graph. To overcome these issues, we decouple feature embedding from propagation using a Spectral GCN formulation, effectively replacing the adjacency matrix with a filtered version of the adjacency matrix in each layer.

2.2 Spectral Graph Convolution

Let $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-1/2}$ denote the symmetrically normalized adjacency matrix of the cell-to-cell similarity graph, and $\boldsymbol{X}$ denote the cell-to-protein abundance matrix used as the node features input to the graph convolutional network. A Spectral GCN encoder [3,7,12,16] propagates the features using a spectral filter $g(\hat{\boldsymbol{A}})$ in each layer of the neural network:

$$\boldsymbol{H}^{(l+1)} = \sigma(g(\hat{\boldsymbol{A}})\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{1}$$

where $\sigma(\cdot)$ is an activation function, e.g., `ReLU`, $\boldsymbol{W}^{(l)}$ a learnable weight matrix, and $\boldsymbol{H}^{(0)} = \boldsymbol{X}$.

In practice, direct computation of $g(\hat{\boldsymbol{A}})$ can be costly for arbitrary $g(.)$. The filter can be defined and evaluated on the eigenspectra of the graph adjacency matrix (Fig 2) or the graph Laplacian, but this is also costly since the computation of the eigendecomposition of very large graphs is cost-prohibitive. Consequently, the filter is usually defined as a polynomial function:

$$g(\hat{\boldsymbol{A}}) = \sum_{k=0}^{K} \theta_k \hat{\boldsymbol{A}}^k \tag{2}$$

Here, K denotes the depth (number of hops) of the propagation on the graph and θ_ks denote the polynomial coefficients used to adjust the relative weight of paths of different length in the graph. A significant benefit to the polynomial representation of filters is the ability to compute $g(\hat{\boldsymbol{A}})\boldsymbol{H}^{(l)}$ as a series of matrix-vector multiplications, without requiring explicit computation of $g(\hat{\boldsymbol{A}})$, which usually is a dense matrix. Furthermore, setting θ_ks as learnable parameters, the GCN can adaptively learn a filter that fits to the training data. Since the initialization of these parameters is essential, an explicit filter function is still required to guide the GCN. For initialization, earlier spectral GCNs use fixed coefficients such as random walks ($\theta_k = \alpha^k$) [12,3]. Later approaches utilize different polynomial bases (e.g., Bernstein [16] or Chebyshev [15] polynomials) without explicit filter alignment. In recent work [6,7], we use polynomial interpolation to explicitly connect a spectral filter to a set of polynomial coefficients, enabling the use of any filter function.

2.3 Polynomial Interpolation of Filter Functions

The low-pass filters employed in this study are defined on the normalized adjacency matrix, whose eigenvalues lie within the range $[-1, 1]$. These filters emphasize the low-frequency components of the graph spectrum, promoting smooth variations of node features across edges. This property aligns with the homophily assumption underlying clustering, where nodes within the same community are expected to share similar attributes.

Given a spectral filter $g(\omega) : [-1, 1] \to \mathbb{R}$ and a desired polynomial degree K, we approximate $g(\omega)$ with a polynomial expansion

$$P(\omega) = \sum_{k=0}^{K} a_k \omega^k, \tag{3}$$

where the coefficients a_k serve as an approximation of the spectral response of the filter. These coefficients are subsequently used to initialize the parameters θ_k in the Spectral GCN encoder (Eq. 2).

To obtain stable and accurate approximations, we sample K spectral points in $[-1, 1]$ using **Chebyshev nodes**:

$$\omega_k = \cos\left(\frac{2k-1}{2K}\pi\right), \quad 1 \le k \le K, \tag{4}$$

We evaluate the filter function at these Chebyshev nodes to obtain samples $g(\omega_k)$ for $1 \le k \le K$.

Vandermonde System. Given the samples, the coefficients of the interpolating polynomial are computed by solving

$$\boldsymbol{V}\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_K \end{bmatrix} = \begin{bmatrix} g(\omega_1) \\ g(\omega_2) \\ \vdots \\ g(\omega_K) \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} 1 & \omega_1 & \dots & \omega_1^K \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_K & \dots & \omega_K^K \end{bmatrix}. \tag{5}$$

Since the Vandermonde matrix $\boldsymbol{V}$ is highly ill-conditioned [29], directly solving linear systems involving $\boldsymbol{V}$ can be numerically unstable. Even performing a QR decomposition, $\boldsymbol{V} = \boldsymbol{Q}_V\boldsymbol{R}_V$, does not guarantee stability:

$$\boldsymbol{a} = \boldsymbol{Q}_V^{\dagger} g, \tag{6}$$

Algorithm 1 Coefficient Computation via Arnoldi Orthonormalization on $\boldsymbol{\Omega}$

Input: bounds l, u, filter g, number of samples r, degree K
Sample r spectral points $\omega_1, \ldots, \omega_r$
Evaluate $g(\omega_1), \ldots, g(\omega_r)$
Construct $\boldsymbol{\Omega} = \mathrm{diag}(\omega_1, \ldots, \omega_r)$
Initialize $\boldsymbol{Q} = \mathbf{1}_r$ ▹ column vector of ones
for $m = 1$ to K **do**
  $\boldsymbol{q} \leftarrow \boldsymbol{\Omega}\boldsymbol{Q}_{:,m}$
  for $j = 1$ to m **do**
    $\boldsymbol{H}(j,m) \leftarrow \boldsymbol{Q}_{:,j}^{\top}\boldsymbol{q}/\sqrt{r}$
    $\boldsymbol{q} \leftarrow \boldsymbol{q} - \boldsymbol{H}(j,m)\boldsymbol{Q}_{:,j}$
  end for
  $\boldsymbol{H}(m+1,m) \leftarrow \|\boldsymbol{q}\|/\sqrt{r}$
  $\boldsymbol{Q} \leftarrow [\boldsymbol{Q}, \boldsymbol{q}/\boldsymbol{H}(m+1,m)]$
end for
$\boldsymbol{a}_A \leftarrow \boldsymbol{Q}^{\dagger}\, g(\omega)$

where $\boldsymbol{Q}_V^{\dagger}$ is the Moore–Penrose pseudoinverse. The fundamental reason for this instability is that Vandermonde matrices are closely related to Krylov subspaces, and when the nodes (or the corresponding eigenvalues) are clustered, the resulting Krylov subspace has nearly linearly dependent columns. As a result, standard QR decomposition algorithms fail to maintain numerical stability in practice [7].

Arnoldi-Orthonormalization. To overcome the aforementioned numerical stability issue, we apply QR decomposition to the diagonalized spectrum $\boldsymbol{\Omega} = \mathrm{diag}(\omega_1, \ldots, \omega_K)$ using Arnoldi orthonormalization. This constructs an orthonormal basis $\boldsymbol{Q}_A$ for the Krylov subspace $\{\boldsymbol{\Omega}^k\mathbf{1}\}_{k=0}^{K}$, yielding:

$$\boldsymbol{a} = \boldsymbol{Q}_A^{\dagger}\, g. \tag{7}$$

Note that, since the ω_ks are real, Arnoldi orthonormalization here can be implemented using Lanczos' algorithm. Specifically, we compute orthonormal matrix $\boldsymbol{Q}_A$, tridiagonal $\boldsymbol{T}$, and an almost-zero matrix $\widetilde{\boldsymbol{O}}$ to satisfy:

$$\boldsymbol{\Lambda}\boldsymbol{Q}_A = \boldsymbol{Q}_A\boldsymbol{T} + \widetilde{\boldsymbol{O}} \tag{8}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{K\times K}$, $\boldsymbol{Q}_A \in \mathbb{R}^{(K+1)\times K}$, $\boldsymbol{T} \in \mathbb{R}^{(K+1)\times(K+1)}$, and $\widetilde{\boldsymbol{O}} \in \mathbb{R}^{(K+1)\times K}$ is all zero except its last column. Then, as shown in [7], we can obtain a QR-decomposition for the Vandermonde matrix $\boldsymbol{V}$ as $\boldsymbol{V}^{(*)} = \boldsymbol{Q}\boldsymbol{R}$, such that $\boldsymbol{V}^{(*)} = \boldsymbol{V}/\|\boldsymbol{e}\|$ and $\boldsymbol{R} = [\boldsymbol{e}_1, \boldsymbol{T}\boldsymbol{e}_1, \cdot, \cdot, \cdot, \boldsymbol{T}^K\boldsymbol{e}_1]$, where $\boldsymbol{e}_1$ denotes the $K+1$-dimensional vector of ones. Consequently, Eq. (7) provides a correct solution for the Vandermonde system of Eq. (5) through Eq. (6). Furthermore, it can be shown that the condition number of $\boldsymbol{Q}_A$ satisfies $\kappa(\boldsymbol{Q}_A^{\dagger}\boldsymbol{Q}_A) \approx 1.01$. [6]. Therefore, Arnoldi orthonormalization, as detailed in Algorithm 1, yields accurate and numerically stable polynomial coefficients for interpolating the filter function.

2.4 Spectral Filters for Clustering Single-Cell Proteomics Data

In this section, we introduce three families of homophilic filters that are implemented in scProfiterole.

1. Random Walk with Restart (RWR) Filter. The RWR filter is defined as:

$$g_{\text{RWR-Filter}}(\hat{\boldsymbol{A}}) = (1-\alpha)(I - \alpha\hat{\boldsymbol{A}})^{-1} = (1-\alpha)\sum_{k=0}^{\infty}\alpha^k\hat{\boldsymbol{A}}^k, \quad \alpha \in (0,1),$$

which is also an infinite sum of adjacency powers, representing repeated neighborhood propagation with restart probability α. While using RWR to in GCNs, the infinite sum is truncated as:

$$g_{\text{RWR Truncated}}(\hat{\boldsymbol{A}}) = (1-\alpha)\sum_{k=0}^{K}\alpha^k\hat{\boldsymbol{A}}^k, \tag{9}$$

Algorithm 2 scProfiterole Convolution Layer (Encoder)

Require: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, edge index $\mathcal{E}$, maximum order K, parameter interval $[-0.9, 0.9]$, filter g
Ensure: Node embedding matrix $\mathbf{H} \in \mathbb{R}^{N \times d'}$

Initialization:
Initialize linear transformation $\mathbf{W} \in \mathbb{R}^{d \times d'}$.
Coefficient computation:
$(c_0, \ldots, c_K) = \texttt{Arnoldi-Orthonormalization}(g, K, [-0.9, 0.9])$ ▹ via Algorithm 1
$\mathbf{c} \leftarrow$ learnable parameters
Propagation:
Apply NN transformation: $\mathbf{X} \leftarrow \mathbf{XW}$.
Initialize output: $\mathbf{H} = c_0\mathbf{X}$.
$\mathbf{X}_{\texttt{prop}} \leftarrow \mathbf{X}$.
for $k = 1$ to K **do**
    $\mathbf{X}_{\texttt{prop}} \leftarrow \hat{\boldsymbol{A}}\, \mathbf{X}_{\texttt{prop}}$ ▹ message propagation
    $\mathbf{H} \leftarrow \mathbf{H} + c_k\, \mathbf{X}_{\texttt{prop}}$
end for
return H

which corresponds to the Kth iterate in the computation of the steady state matrix via power iteration. Therefore, the truncated sum may not provide a good approximation to $g_{\text{RWR-Filter}}(\hat{\boldsymbol{A}})$ for small values of K (which is often the case in GCN applications).

In contrast, the interpolated version of RWR uses the $K + 1$ polynomial coefficients (Eq. 2) that interpolate (Eq. 3) the following filter function, as computed by Arnoldi orthonormalization:

$$g_{\text{RWR-Interpolated}}(\omega) = (1 - \alpha)\frac{\alpha}{1 - \omega}. \tag{10}$$

2. Heat Kernel Filter. While the RWR filter is simple and interpretable, it is concentrated around a narrow band near the largest eigenvalues of $\hat{\boldsymbol{A}}$ (Fig. 2, left). Different values of the parameter $alpha$ provide little variation within this narrow band, making limiting the flexibility of the RWR-filter in dealing with the sparse and noisy nature of the cell-to-cell similarity graph in scProteomics. Motivated by this insight, we also implement heat kernels in scProfiterole.

Let $\mathcal{L} = \boldsymbol{I} - \hat{\boldsymbol{A}}$ denote the graph Laplacian. The heat kernel [22] is defined as:

$$g_{\text{Heat}}(\mathcal{L}) = e^{-T\mathcal{L}} = e^{-T(\boldsymbol{I} - \hat{\boldsymbol{A}})} = e^{-T}\, e^{T\hat{\boldsymbol{A}}} = g_{\text{Heat}}(\hat{\boldsymbol{A}}). \tag{11}$$

This can be thought of as a continuous random walk on the graph, where the parameter T specifies the duration of the walk. Kloster and Gleich [22] approximate the heat kernel using Taylor series:

$$g_{\text{Heat-Approximated}}(\hat{\boldsymbol{A}}) = e^{-T} \sum_{k=0}^{K} \frac{(T\hat{\boldsymbol{A}})^k}{k!}. \tag{12}$$

Since Taylor series approximates the function around a fixed point, numerical polynomial interpolation may provide a more accurate approximation to the heat kernel. As above, we define the interpolated version of Heat kernel as the polynomial that uses the $K + 1$ polynomial coefficients (Eq. 2) that interpolate (Eq. 3) the following filter function, as computed by Arnoldi orthonormalization:

$$g_{\text{Heat-Interpolated}}(\omega) = e^{-T + T\omega}, \tag{13}$$

3. Beta Kernel Filter. Beta distribution is commonly used as a wavelet basis for computer vision applications, but it also demonstrated promise as a kernel for anomaly detection on graphs [38]. While the general form of beta kernels can be used to define any type of filter on graph spectra, the zeroth-order Beta kernel provides a low-pass filter that can be directly represented as a polynomial of order K, where K serves a parameter (Fig 2, center). For this reason,

Table 1: **scProteomics datasets used in our experiments.**

Name	# Cells	# Proteins	Cell type (#)
Scope2_Specht	1490	3042	Monocytes (394) and Macrophages (1096)
N2	108	1068	C10 (36), RAW (36), SVEC (36)
nanoPOTS	61	1225	C10(19), RAW(22), SVEC(20)

zeroth order Beta kernel offers a desirable alternative for low-pass filter design, since it does not require truncation, approximation, or interpolation.

For a given K, the zeroth order Beta kernel is defined as follows:

$$g_{\text{Beta}} = (K+1)\left(\frac{I+\hat{A}}{2}\right)^{K}. \tag{14}$$

Observe that coefficients that correspond to this polynomial are the binomial coefficients normalized by $2^K/(K+1)$. We implement the beta kernel by directly incorporating these coefficients as the initial values of the coefficients in the spectral filter (Eq. 2). The proposed encoder details is given in Algorithm 2, full mathematical procedure can be found in Appendix subsection A.

In summary, we introduce a polynomial interpolation approach for spectral graph filters, enabling scalable and stable low-pass filtering of node features in GCN-Encoders. Filter coefficients are computed using Chebyshev nodes and stabilized via Arnoldi orthonormalization, overcoming the ill-conditioning of standard Vandermonde systems. We implement three homophilic filters: (i) Random Walk with Restart (RWR) for repeated neighborhood propagation, (ii) Heat Kernel for continuous graph diffusion, and (iii) zeroth-order Beta Kernel as a direct low-pass polynomial. For RWR and Heat kernels, polynomial interpolation ensures accurate and stable approximation, while the Beta kernel provides a straightforward low-pass design, collectively enhancing node feature smoothness and cluster coherence in single-cell proteomics data.

3 Results

In this section, we first detail the datasets and experimental setup. We then present the performance of the spectral graph filters implemented by scProfiterole in the context of clustering scProteomics data and cell type annotation. Subsequently, we investigate the differences between the spectral filters in terms of how they initialize the polynomial coefficients, how initialization influences the learned coefficients, and what filter functions correspond to the learned coefficients. We then discuss the insights derived from these results on the design of spectral filters that are best suited to the clustering of cell-to-cell similarity graphs derived from scProteomics data.

3.1 Datasets and Experimental Setup

We evaluate scProfiterole in the context of clustering cell-to-cell similarity graphs derived from scProteomics data and cell type annotation.

Datasets and Preprocessing of scProteomics Data. The datasets we use for this purpose are shown in Table 1. To comprehensively compare the performance of the spectral filters implemented in scProfiterole and investigate the effect of parameters, we use the Scope2_Specht dataset [35], profiling 3,042 proteins across 1,490 cells. To test robustness against batch effects and assess the generalizibility of our conclusions to other datasets, we also obtain a second dataset by integrating two relatively smal datasets, N2 [10] (108 cells, 1,068 proteins) and nanoPOTS [10] (61 cells, 1,225 proteins). We use the 762 overlapping proteins between these two datasets to construct a shared protein expression data matrix, and construct the cell-to-cell similarity graph using this integrated data matrix. To obtain the cell-to-cell similarity graph, we utilize the framework introduced by scPROTEIN [25], performing peptide-level uncertainty estimation via multitask heteroscedastic regression and uncertainty-weighted aggregation to obtain noise-reduced protein abundances, . followed by the computation of the pairwise correlations between the protein expression profiles of the cells.

Performance Criteria. To assess the performance of the filters in clustering and cell-type annotation, we use four standard performance criteria: Adjusted Rand Index (ARI) [17], Average Silhouette Width (ASW) [33], Normalized

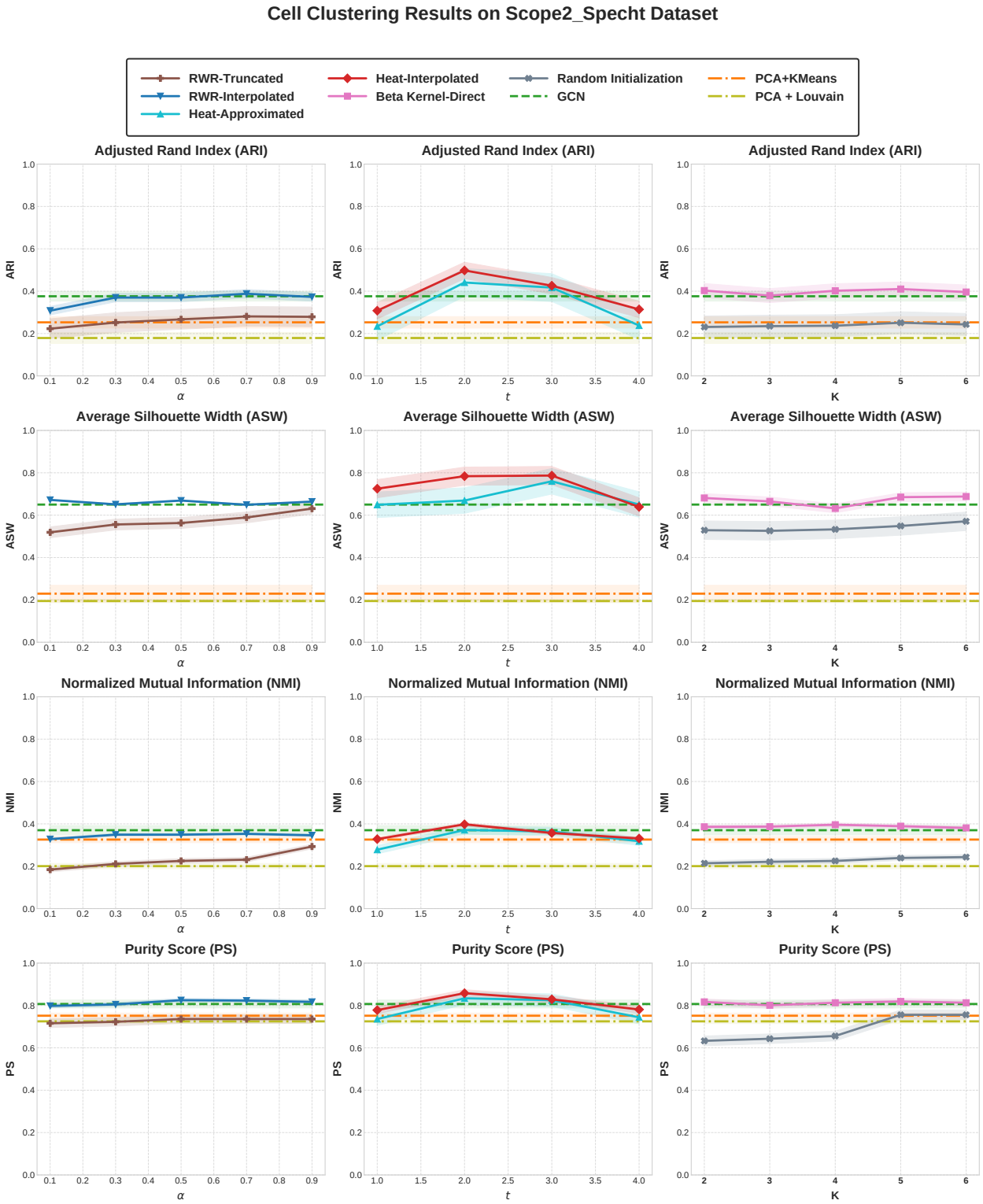

Fig. 3: **Clustering performance of the spectral filters implemented in** scProfiterole **on the** `Scope2_Specht` **dataset.** Four performance criteria are shown as a function of restart probability (α) for random-walk based filters (Left), diffusion time (t) for Heat kernel filters (Middle), and polynomial degree (K) for the Beta filters. For random walk and heat kernel, $K = 6$. Plot shows Mean ± 95% Confidence Interval over eight different seed runs.

Mutual Information (NMI) [11], and Purity Score (PS) [28]. The definitions of these measures are provided in Appendix B. For all of these measures, higher values indicate more desirable performance.

Parameter Settings. The families of spectral filters we implement have one parameter in common: The degree K of the polynomial that is used to implement the filter. While this is the sole parameter for the Beta kernels, Random walk filters and Heat kernels each have one additional parameter: Restart probability (α) for Random walk, diffusion time (t) for Heat kernels. In our first round of experiments we fix $K = 6$ for the Random walk and Heat kernel filters, while we vary K from 2 to 6 for the Beta kernels. We choose this value of K as we expect the cell-to-cell similarity graphs to be small world networks (thus a smaller K can be desirable) and we would like to maximize the expressive power of the polynomial approximation/interpolation (thus a larger K can be desirable). Regardless, we perform additional experiments for values of K up to 24 to fully characterize the effect of the polynomial degree on the performance of spectral filters. For random walks, we use five equispaced values of α in the range $[0.05, 0.95]$. For the heat kernel, we consider integer values of diffusion time T ranging from 1 to 5. Finally, for the primary experiments reported here, we set the number of dimensions for cell embeddings to $D = 64$, but we also investigate the effect of D for values ranging from 32 to 256.

3.2 Performance Evaluation

We first evaluate the clustering performance of the Random Walk with Restart (RWR) filters, Heat Kernel filters, and Beta Kernel filters on the `Scope2_Specht` dataset. For the RWR filter, we consider the truncated (standard) and interpolated (proposed) versions. For the Heat kernel, we consider the approximated (standard) vs. interpolated (proposed) versions. Since Beta Kernel naturally employs a finite polynomial basis, we consider its direct implementation. As baseline models, we consider several approaches: (i) embeddings obtained from a GCN encoder that performs convolution using the adjacency matrix of the cell–cell similarity graph, (ii) random initialization (without filter guidance), (iii) PCA applied to the feature matrix to obtain 128-dimensional embeddings followed by K-means clustering, and (iv) Louvain clustering with KNN = 15 applied to the PCA embeddings. The results of this analysis are shown in Figure 3 in which Mean ± 95% Confidence Interval in eight different seed runs.

Heat kernels deliver the best clustering performance. As seen in Figure 3, the Heat kernel based spectral filters outperform other filters, delivering optimal performance for $T \in [2, 3]$. For $t = 2$, as compared to the baseline GCN that uses the adjacency matrix for convolution, Interpolated Heat kernel provides 29.7% improvement in ARI, 9.1% improvement in ASI, 8.1% improvement in NMI, and 3.7% improvement in PS. RWR filter delivers inferior performance as compared to the baseline, but its performance is improved with increasing polynomial degree (K), as would be expected. Interpolation makes RWR competitive with the baseline and the effect of K vanishes with interpolation. Similarly, the Beta kernel delivers performance that is competitive with the baseline, with little dependence on K. These results show that the performance of spectral graph filters in clustering scProteomics data depends highly on the choice of the spectral filter. While Interpolated Heat kernels provide consistent and considerable improvement over the baseline, other filters either fail to deliver meaningful improvement or have a degrading effect on clustering performance.

In contrast, the non-graph baselines remain clearly inferior. Both PCA followed by K-means and PCA followed by Louvain clustering yield substantially lower scores across all evaluation metrics, indicating that purely feature-based dimensionality reduction without effective graph modeling fails to capture the intrinsic cell–cell relationships. In particular, PCA+Louvain exhibits the weakest performance, suggesting that community detection on a KNN graph constructed in the reduced feature space is insufficient for this dataset. Random initialization performs slightly better than PCA-based clustering in some metrics, yet remains far below the proposed spectral filter-based methods, highlighting that meaningful graph filtering is essential for obtaining discriminative embeddings. Overall, these results further confirm that the observed performance gains stem from the choice of spectral graph filter rather than from the clustering algorithm or dimensionality reduction alone.

Interpolation via Arnoldi orthonormalization enhances spectral filter performance. As seen in Figure 3, for both the RWR and Heat kernel filters, we observe considerable performance gap between the interpolated vs. truncated (for RWR) or approximated (Heat kernel) filters. While the performance of the RWR filter is improved with increasing order of the polynomial representation of the filter (K), interpolation removes this dependency. This observation suggests that interpolation of spectral filters enables working with polynomials of smaller order, capturing higher order relationships in the graph by considering shorter paths than those needed by truncated polynomials. Interpolated filters accomplish this by choosing the polynomial coefficients (weights assigned to path lengths) to approximate the true filter with fidelity. Interpolation also results in drastic improvement in the performance of the Heat kernel filter, making Heat kernels the best choice of spectral filters for clustering scProteomics data. For Heat kernel based filters, the dependency on diffusion

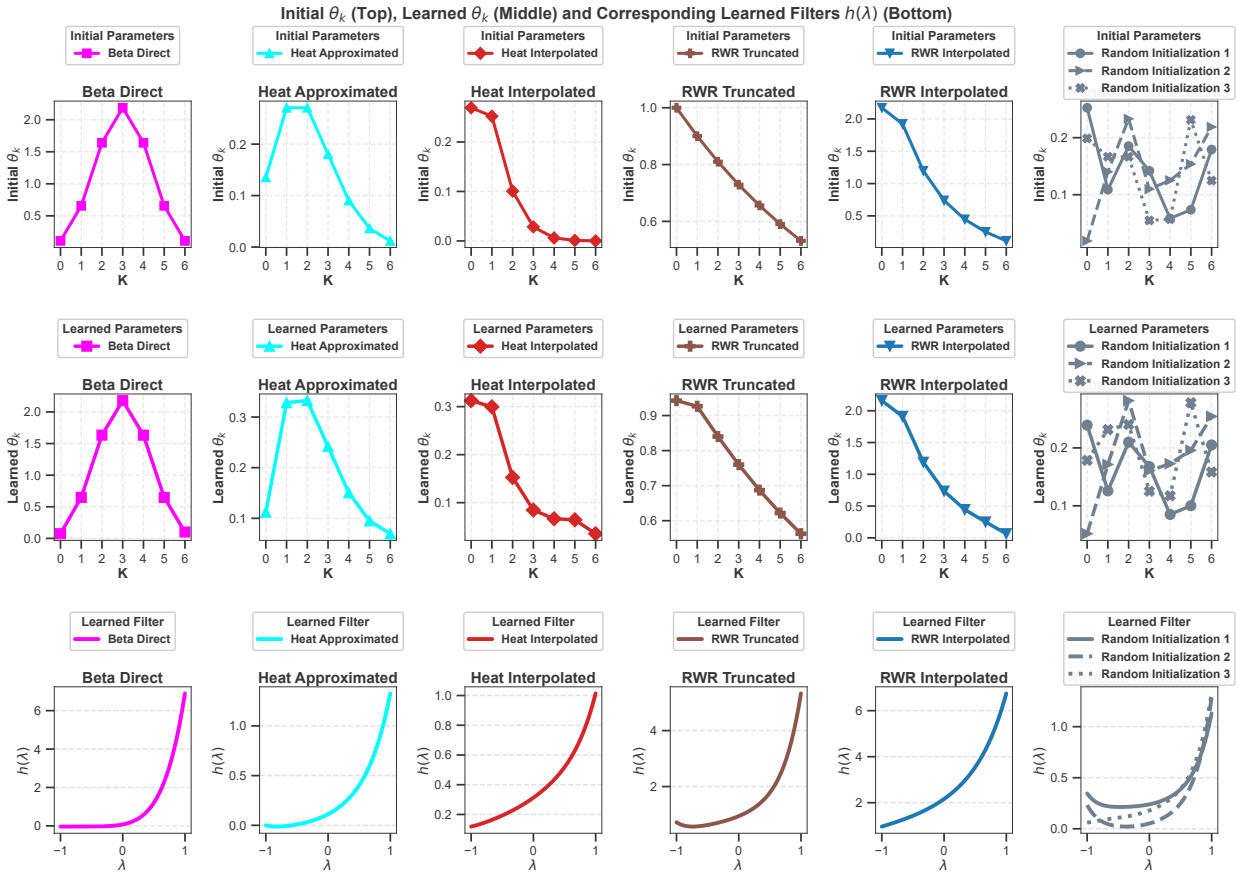

Fig. 4: **Learned spectral coefficients and corresponding filters on the** `Scope2_Specht` **dataset.** Comparison of initial and learned polynomial coefficients θ_k (top and middle rows, respectively) and filter function $h(\lambda)$ corresponding to the learned coefficients (bottom row) across five spectral filters implemented in scProfiterole: Beta Kernel (Direct), Heat Kernel (Approximated and Interpolated), and Random Walk with Restart (Truncated and Interpolated). In these experiments, $K = 6$ for all filters, $\alpha = 0.9$ for the RWR filter, and $t = 2$ for Heat kernel.

time T is consistent between the approximated and interpolated versions, where clustering performance reaches its peak at $T = 2$ and declines rapidly for $T > 3$. These results suggest that the ability to interpolate spectral filters using a numerically stable framework enables spectral filters to realize their potential in the context of clustering scProteomics data.

The learned filter strictly depends on the initialization of the polynomial coefficients. In Fig. 4, the initial values of the polynomial coefficients, the learned values of the polynomial coefficients, and the filter function that corresponds to the learned polynomial are shown for the setting that results in the optimal performance for each filter family on the `Scope2_Specht` dataset. Comparison of the first two rows of this figure suggests that the learned polynomial coefficients heavily depend on their initial values. This result suggests that the optimization of the polynomial coefficients alongside neural network parameters may be hampering the search in the coefficient space. To this end, at the absence of dedicated learning algorithms that optimize polynomial coefficients more effectively, the choice of the spectral filter for initialization remains a significant hyperparameter. Taken together, Figures 3 and 4 suggest that interpolated filters deliver superior performance by initializing the polynomial coefficients more effectively. This sensitivity is further evidenced by the random initialization experiments shown in Fig. 4. Different random initializations lead to noticeably different learned polynomial coefficients and, consequently, to substantially different learned filter shapes, despite using the same model and training protocol. This effect is further illustrated by the random initialization cases in Fig. 4, where different random starts lead to markedly different learned coefficients and filter shapes, indicating convergence to different local minima. This instability helps explain the weaker and less consistent random initialization performance observed in Fig. 3.

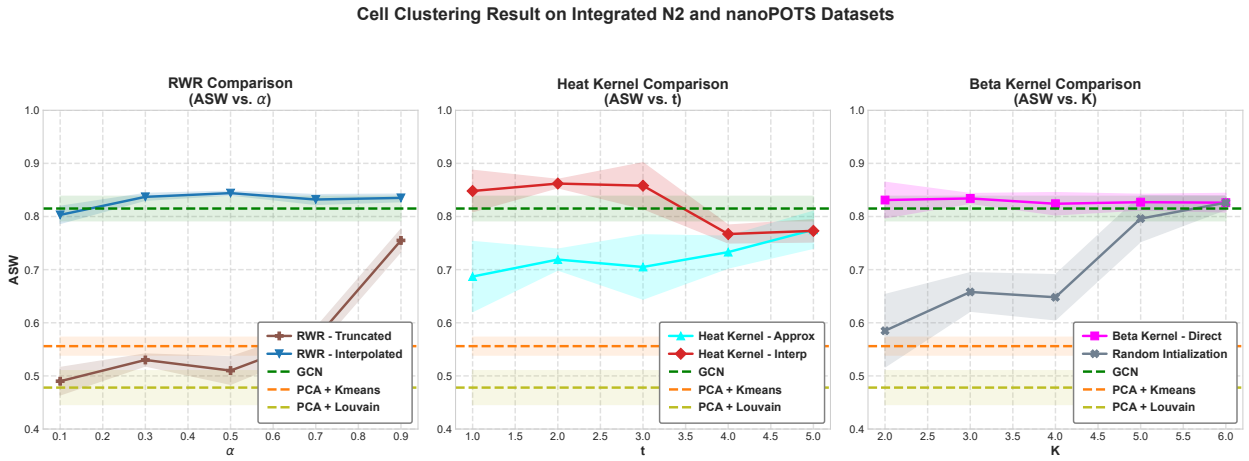

Fig. 5: **Cell clustering performance of the spectral filters on the integrated `N2` and `nanoPOTS` datasets.** ASW scores are reported for Truncated vs. Interpolated Random walk filters as a function of restart probability α (left), Beta kernel as a function of the order K of the polynomial filter (center), and Approximated vs. Interpolated Heat kernel as a function of diffusion time T (right). The figure shows Mean ± 95% Confidence Interval over eight different seed runs.

Low-pass filters that are effective clustering scProteomics data provide a broader view of the eigenspectra. The last row of Fig. 4 shows the filter function that corresponds to the learned polynomial coefficients for each filter family. While all functions represent low-pass filters (recall that these functions are defined on the eigenvalues of the adjacency matrix), the curves for the interpolated Heat kernel and Random walk filters represent a unique pattern. Unlike the other filters that suppress negative eigenvalues, these filters permit negative eigenvalues, with preference to eigenvalues with smaller magnitude. For positive eigenvalues of the adjacency matrix, these filters assign more importance to eigenvalues with larger magnitude, but the rise of the curve is not as sharp or pronounced as other filters. Inspecting the initial and learned polynomial coefficients in the first and second rows of the figure, we observe that these filters accomplish this by assigning considerable more weight to the first two polynomial coefficients.

The key observations are replicated on a second dataset. To assess the generalization of our observations and the robustness of the spectral filters to heterogeneous data sources, we also evaluate the clustering performance of spectral filters on the integrated `N2` and `nanoPOTS` datasets. Fig. 5 shows the Average Silhouette Width (ASW) of the clustering delivered by each filter family as a function of its key parameters. Consistent with the `Scope2_Specht` dataset, we observe that the interpolated Heat kernel delivers best clustering performance, with peak performance at $T = 2$. We also observe that the interpolated filters outperform truncated (for Random walk) and approximated (for Heat kernel) versions of the filters. The cell clustering provided by the truncated Random walk filter is of lower quality than that obtained using the baseline, but the quality of clustering improves with increasing order of the polynomial filter. Arnoldi orthonormalization based interpolation helps the Random walk filter deliver better clustering than the baseline, while also removing the dependency on K. The cell clustering provided by the Beta kernel is competitive with the baseline, with no clear dependency on K. We also compare our methods against PCA+Kmeans, PCA+Louvain, and random initialization. Both PCA-based baselines are clearly inferior to the spectral methods and the GCN baseline, with PCA+Louvain performing worst. Random initialization exhibits high variance and only becomes competitive at larger K, underling that the observed gains stem from the interpolated spectral filters rather than from dimensionality reduction or random bases.

The effect of parameters and the visualization of cell clustering. The cell clusters obtained by the spectral filters are visualized in Appendix Figure 7. These plots show that the latent representations of cells from the `N2` and `nanoPOTS` datasets are better separated by cell type and less affected by batch effects when spectral filters are interpolated. In particular, interpolated versions for the Random-walk and Heat kernel filters yield tighter, more coherent clusters, whereas truncated and approximated filters show more overlap between cell types. Appendix Figure 6 investigates whether using a higher-order polynomial filter would result in improved clustering performance. As seen in the figure, the order of the polynomial filter does not significantly affect clustering performance for the interpolated filters. Appendix Figure 8 shows the effect of the number of cell embedding dimensions on clustering performance for each family of filters. Interpolated Heat kernels again deliver the best performance across different embedding dimensions, showing that the polynomial coefficients set by this filter robustly transform graph topology to bring the cluster homophily forward.

4 Conclusion

We present scProfiterole, a spectral graph contrastive learning framework tailored to clustering single-cell proteomics data. By utilizing learnable low-pass spectral filters including random walk, heat, and beta kernels, and efficient Arnoldi orthonormalization-based polynomial interpolation, scProfiterole overcomes challenges of graph sparsity, noise, and oversmoothing without requiring deep network architectures. Experiments on state-of-the-art scProteomics datasets demonstrate that scProfiterole enhances clustering and cell type annotation, with heat kernel initialization providing the most consistent improvements. Overall, scProfiterole integrates the interpretability of spectral graph theory with the flexibility of contrastive learning, providing a scalable and principled foundation for robust representation learning in high-dimensional, noisy single-cell proteomic data.

Acknowledgments. This work was supported in part by US National Institutes of Health (NIH) grant R01-LM012980 from the National Library of Medicine. The authors would like to thank Kaan Yorgancıoğlu (CWRU), Kasey Wei (CWRU), Willis Erdman (CWRU), Aidan Leblanc (CWRU), Abdelkader Baggag (Qatar Computing Research Institute), and Ananth Grama (Purdue) for many useful discussions.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Boekweg, H., Payne, S.H.: Challenges and opportunities for single-cell computational proteomics. Molecular & Cellular Proteomics **22**(4), 100518 (2023)
2. Buccitelli, C., Selbach, M.: mrnas, proteins and the emerging principles of gene expression control. Nature Reviews Genetics **21**, 630–644 (2020)
3. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: 9th International Conference on Learning Representations, ICLR 2021 (2021)
4. Ciortan, M., Defrance, M.: Gnn-based embedding for clustering scrna-seq data. Bioinformatics **38**(4), 1037–1044 (2022)
5. Cong, W., Ramezani, M., Mahdavi, M.: On provable benefits of depth in training graph convolutional networks. Advances in Neural Information Processing Systems **34**, 9936–9949 (2021)
6. Coskun, M., Baggag, A., Koyutürk, M.: Arnoldigcl: Graph contrastive learning via learnable arnoldi-based guided spectral chebyshev polynomial filters. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2. pp. 380–391 (2025)
7. Coşkun, M., Grama, A., Koyutürk, M.: Generalized learning of coefficients in spectral graph convolutional networks. In: 2024 IEEE International Conference on Knowledge Graph (ICKG). pp. 25–32. IEEE (2024)
8. Coşkun, M., Koyutürk, M.: Node similarity-based graph convolution for link prediction in biological networks. Bioinformatics **37**(23), 4501–4508 (2021)
9. Cowen, L., Ideker, T., Raphael, B.J., Sharan, R.: Network propagation: A universal amplifier of genetic associations. Nature Reviews Genetics **18**(9), 551–562 (2017)
10. Dou, M., Clair, G., Tsai, C.F., Xu, K., Chrisler, W.B., Sontag, R.L., Zhao, R., Moore, R.J., Liu, T., Pasa-Tolic, L., et al.: High-throughput single cell proteomics enabled by multiplex isobaric labeling in a nanodroplet sample preparation platform. Analytical chemistry **91**(20), 13119–13127 (2019)
11. Estévez, P., Tesmer, M., Pérez, C., Zurada, J.: Normalized mutual information feature selection. IEEE Transactions on Neural Networks **20**(2), 189–201 (2009)
12. Gasteiger, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019)
13. Gatto, L., Aebersold, R., Cox, J., Demichev, V., Derks, J., Emmott, E., Franks, A.M., Ivanov, A.R., Kelly, R.T., Khoury, L., et al.: Initial recommendations for performing, benchmarking and reporting single-cell proteomics experiments. Nature methods **20**(3), 375–386 (2023)
14. Guo, T., Steen, J.A., Mann, M.: Mass-spectrometry-based proteomics: from single cells to clinical applications. Nature **638**(8052), 901–911 (2025)
15. He, M., Wei, Z., Wen, J.R.: Convolutional neural networks on graphs with chebyshev approximation, revisited. Advances in neural information processing systems **35**, 7264–7276 (2022)
16. He, M., Wei, Z., Xu, H., et al.: Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. Advances in neural information processing systems **34**, 14239–14251 (2021)
17. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**(1), 193–218 (1985)
18. Kambhampati, S., D'Alessio, L., Grab, F., Fleming, S., Chen, F., Babadi, M.: Tissuemosaic enables cross-sample differential analysis of spatial transcriptomics datasets through self-supervised representation learning. In: Proceedings of the 29th Annual International Conference on Research in Computational Molecular Biology (RECOMB). Springer (2025)
19. Kavran, A.J., Clauset, A.: Denoising large-scale biological data using network filters. BMC Bioinformatics **22**, 157 (2021)
20. Kenlay, H., Thanou, D., Dong, X.: Interpretable stability bounds for spectral graph filters. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 5388–5397. PMLR (18–24 Jul 2021)
21. Kipf, T.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
22. Kloster, K., Gleich, D.F.: Heat kernel based community detection. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1386–1395 (2014)
23. Li, M., Coşkun, M., Koyutürk, M.: Topological-similarity based canonical representations for biological link prediction. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2024)
24. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
25. Li, W., Yang, F., Wang, F., Rong, Y., Liu, L., Wu, B., Zhang, H., Yao, J.: scprotein: a versatile deep graph contrastive learning framework for single-cell proteomics embedding. Nature Methods **21**(4), 623–634 (2024)
26. Liang, S., Tang, J., Wang, G., Ma, J.: Steamboat: Attention-based multiscale delineation of cellular interactions in tissues. In: Research in Computational Molecular Biology (RECOMB 2025). Lecture Notes in Computer Science, vol. 15647, pp. 432–436. Springer (2025)
27. Liu, T., Brouwer, E.D., Kuo, T., Diamant, N., Missarova, A., Wang, H., Hao, M., Bravo, H.C., Scalia, G., Regev, A., Heimberg, G.: Learning multi-cellular representations of single-cell transcriptomics data enables characterization of patient-level disease states. In: Proceedings of the 29th Annual International Conference on Research in Computational Molecular Biology (RECOMB). Springer (2025)

28. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2008)
29. Pan, V.Y.: How bad are vandermonde matrices? SIAM Journal on Matrix Analysis and Applications **37**(2), 676–694 (2016)
30. Perl, K., Ushakov, K., Pozniak, Y., Yizhar-Barnea, O., Bhonker, Y., Shivatzki, S., Geiger, T., Avraham, K.B., Shamir, R.: Reduced changes in protein compared to mrna levels across non-proliferating tissues. BMC Genomics **18**, 305 (2017)
31. Prabahar, A., Zamora, R., Barclay, D., Yin, J., Ramamoorthy, M., Bagheri, A., Johnson, S.A., Badylak, S., Vodovotz, Y., Jiang, P.: Unraveling the complex relationship between mrna and protein abundances: a machine learning-based approach for imputing protein levels from rna-seq data. NAR Genomics and Bioinformatics **6**(1), lqae019 (2024)
32. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903 (2019)
33. Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**(1), 53–65 (1987)
34. Specht, H., Emmott, E., Petelski, A.A., Huffman, R.G., Perlman, D.H., Serra, M., Kharchenko, P., Koller, A., Slavov, N.: Single-cell proteomic and transcriptomic analysis of macrophage heterogeneity using scope2. Genome Biology **22**(1), 50 (2021)
35. Sticker, A., Goeminne, L., Martens, L., Clement, L.: Robust summarization and inference in proteome-wide label-free quantification. Molecular & Cellular Proteomics **19**(7), 1209–1219 (2020)
36. Svensson, V., Vento-Tormo, R., Teichmann, S.A.: Exponential scaling of single-cell rna-seq in the past decade. Nature protocols **13**(4), 599–604 (2018)
37. Tang, H., Liu, Y.: Towards understanding generalization of graph neural networks. In: International Conference on Machine Learning. pp. 33674–33719. PMLR (2023)
38. Tang, J., Li, J., Gao, Z., Li, J.: Rethinking graph neural networks for anomaly detection. In: International conference on machine learning. pp. 21076–21089. PMLR (2022)
39. Wang, J., Ma, A., Chang, Y., Gong, J., Jiang, Y., Qi, R., Wang, C., Fu, H., Ma, Q., Xu, D.: scgnn is a novel graph neural network framework for single-cell rna-seq analyses. Nature communications **12**(1), 1882 (2021)
40. Yang, W., Wang, P., Luo, M., Cai, Y., Xu, C., Xue, G., Jin, X., Cheng, R., Que, J., Pang, F., et al.: Deepcci: a deep learning framework for identifying cell–cell interactions from single-cell rna sequencing data. Bioinformatics **39**(10), btad596 (2023)

Appendix

scProfiterole: Clustering of Single-Cell Proteomic Data UsingGraph Contrastive Learning via Spectral Filters

Mustafa Coşkun, Filipa Blasco Lopes, Pınar Tolunay Kubilay, Mark R. Chance, Mehmet Koyutürk

A Mathematical Details of scProfiterole

A.1 Data augmentation.

Contrastive learning seeks to learn invariant representations by comparing similar and dissimilar data pairs. To construct similar (positive) pairs, we generate multiple augmented views of the input graph using two augmentation strategies: *edge dropping* [32] and *feature masking*. Although a variety of contrastive mechanisms exist—such as topology-aware graph views proposed in [8,23] in addition to edge dropping—we focus exclusively on *edge dropping* [32] and *feature masking* to maintain consistency with the scPROTEIN framework. Exploring these alternative augmentation techniques is left for future work.

Edge dropping. Given an edge-dropping ratio p_{de}, we randomly remove edges from E. Specifically, we sample an indicator matrix $\boldsymbol{R} \in \mathbb{R}^{N\times N}$ such that

$$\boldsymbol{R}_{ij} \sim \begin{cases} \text{Bernoulli}(1-p_{\text{de}}), & \boldsymbol{A}_{ij}=1, \\ 0, & \boldsymbol{A}_{ij}=0, \end{cases}$$

and obtain the perturbed adjacency matrix

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} \odot \boldsymbol{R},$$

where $\odot$ denotes the Hadamard product.

Feature masking. To mask node features, we sample a masking vector $\boldsymbol{M} \in \mathbb{R}^{N\times 1}$ with entries

$$\boldsymbol{M}_i \sim \text{Bernoulli}(1-p_{\text{mf}}),$$

where p_{mf} is the masking probability. The masked feature matrix is then given by

$$\widetilde{\boldsymbol{X}} = \begin{bmatrix} \mathbf{x_1} \odot \boldsymbol{M} \\ \mathbf{x_2} \odot \boldsymbol{M} \\ \vdots \\ \mathbf{x_N} \odot \boldsymbol{M} \end{bmatrix},$$

where $[\cdot;\cdot]$ denotes row-wise vector concatenation.

In each training iteration, these augmentation procedures produce two correlated graph views,

$$\mathcal{G}_1 = (\mathcal{V}, \widetilde{\mathcal{E}}_1, \widetilde{\boldsymbol{X}}_1), \qquad \mathcal{G}_2 = (\mathcal{V}, \widetilde{\mathcal{E}}_2, \widetilde{\boldsymbol{X}}_2),$$

which are used for contrastive learning.

A.2 Arnoldi-orthonormalization–based spectral graph encoder.

Given the augmented graph views $\mathcal{G}_1$ and $\mathcal{G}_2$, we learn cell embeddings via node-level graph contrastive learning using an Arnoldi-guided spectral encoder. The standard GCN [21] serves as the baseline encoder, where an L-layer GCN updates node representations through neighborhood aggregation:

$$\boldsymbol{Z}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{15}$$

with $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-1/2}$ denoting the normalized adjacency, $\boldsymbol{W}^{(l)}$ the trainable weights, and $\sigma(\cdot)$ a (parametric) ReLU. However, as illustrated in Fig. 1, deep GCNs suffer from over-smoothing, motivating the use of spectral propagation.

A spectral GCN encoder [3,7,12,16] applies a spectral filter $g(\hat{\boldsymbol{A}})$ at each layer:

$$\boldsymbol{H}^{(l+1)} = \sigma\Big(g(\hat{\boldsymbol{A}})\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\Big), \qquad \boldsymbol{H}^{(0)} = \boldsymbol{X}. \tag{16}$$

Direct computation of $g(\hat{\boldsymbol{A}})$ via eigendecomposition is impractical on large graphs; therefore the filter is implemented as a K-th order polynomial:

$$g(\hat{\boldsymbol{A}}) = \sum_{k=0}^{K} \theta_k \hat{\boldsymbol{A}}^k, \tag{17}$$

allowing efficient evaluation through iterative matrix–vector multiplications. The coefficients θ_k become learnable filter parameters. Earlier spectral GCNs used fixed choices (e.g., random-walk $\theta_k = \alpha^k$ [12,3]) or predefined bases (Bernstein [16], Chebyshev [15]). Our recent work [6,7] instead employs polynomial interpolation to explicitly align any desired spectral filter with its polynomial coefficients, enabling Arnoldi-guided spectral learning (Algorithm 1). Note that the main text introduces three families of filters $g(\cdot)$: (i) `RWR-Truncated` and `RWR-Interpolated` (Eq. 9, 10), (ii) `Heat-Approximated` and `Heat-Interpolated` (Eq. 12, 13), and (iii) the `Beta-Direct` kernel (Eq. 14).

For both augmented views, we use a weight-sharing Arnoldi-based spectral GNN (Algorithm 2) to obtain embeddings $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, with all layer dimensionalities kept consistent across views.

A.3 Node-level graph contrastive learning.

The embeddings $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ from the two augmented views are passed through a shared projection head $\phi(\cdot)$ (a two-layer MLP) that maps them into a common latent space where the contrastive loss is computed. For each node i, its projected embeddings $\phi(\boldsymbol{h}_{1i})$ and $\phi(\boldsymbol{h}_{2i})$ form a positive pair, while all other node embeddings act as negatives.

We use cosine similarity $\cos(\cdot,\cdot)$ and the InfoNCE loss. For a positive pair $(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i})$, the node-level contrastive loss is

$$\ell(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i}) = -\log \frac{\exp(\theta(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i})/\tau)}{\exp(\theta(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i})/\tau) + \sum_{j\neq i}\big[\exp(\theta(\boldsymbol{h}_{1i}, \boldsymbol{h}_{1j})/\tau) + \exp(\theta(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j})/\tau)\big]}, \tag{18}$$

where $\theta(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i}) = \cos(g(\boldsymbol{h}_{1i}), g(\boldsymbol{h}_{2i}))$ and τ is a temperature parameter. Here, $(\boldsymbol{h}_{1i}, \boldsymbol{h}_{1j})$ are within-view negatives, and $(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j})$ are cross-view negatives.

Since the two views are symmetric, the final node-level contrastive loss is

$$\mathcal{L}_{\text{node}} = \frac{1}{2N}\sum_{i=1}^{N}\big[\ell(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i}) + \ell(\boldsymbol{h}_{2i}, \boldsymbol{h}_{1i})\big]. \tag{19}$$

Minimizing this loss pulls positive pairs together while pushing negative pairs apart, preserving biological variability across cells and implicitly correcting batch effects, ultimately yielding more robust node embeddings.

A.4 Alternating topology–attribute denoising.

Single-cell proteomic data contain substantial noise from sample preparation and MS quantification. To obtain noise-resilient embeddings, we adopt an alternating denoising strategy consisting of (i) attribute denoising and (ii) topology-based contrastive learning.

Attribute denoising. To reduce noise in the proteomic profiles, we use prototype contrastive learning. Prototypes lying far from cluster boundaries are more robust to noise [13], so we leverage them to guide embedding refinement. At each epoch, we cluster the current embeddings (without augmentation) into K groups via k-means, yielding cluster assignments cluster_i and cluster centers $\{c_1, \ldots, c_K\}$. Each prototype c_k is the mean embedding of nodes in cluster k:

$$c_k = \frac{1}{\text{Num}_k}\sum_{\text{cluster}_i = k} \boldsymbol{h}_i, \tag{20}$$

where Num_k is the number of nodes assigned to cluster k.

We then update all embeddings using the prototype-level contrastive loss

$$\mathcal{L}_{\text{proto}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\boldsymbol{h}_i \cdot c_{\text{cluster}_i}/\tau)}{\sum_{k=1}^{K}\exp(\boldsymbol{h}_i \cdot c_k/\tau)}. \tag{21}$$

Overall loss. The final objective combines node-level contrastive learning (based on the augmented embeddings $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$) with prototype denoising:

$$\mathcal{L} = \mathcal{L}_{\text{node}} + \gamma\, \mathcal{L}_{\text{proto}}, \tag{22}$$

where γ is a balancing coefficient (set to 0.05 in our experiments). The model is optimized using Adam with a learning rate of 10^{-3}.

B Evaluation Metrics

The metrics are formally defined as follows:

Adjusted Rand Index (ARI):

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}$$

where n_{ij} denotes the number of cells assigned to both true cluster i and predicted cluster j, and a_i, b_j represent the respective cluster marginals.

Average Silhouette Width (ASW):

$$\text{ASW} = \frac{1}{N} \sum_i \frac{p(i) - q(i)}{\max\{p(i), q(i)\}}$$

where $q(i)$ is the average intra-cluster distance for cell i, $p(i)$ is the smallest average inter-cluster distance, and N is the total number of cells.

Normalized Mutual Information (NMI):

$$\text{NMI} = \frac{MI(\hat{Y}; Y)}{\sqrt{H(\hat{Y})H(Y)}}$$

where $MI(\hat{Y}; Y)$ denotes the mutual information between predicted cluster assignments $\hat{Y}$ and true labels Y, and $H(\cdot)$ represents Shannon entropy.

Purity Score (PS):

$$\text{PS} = \frac{1}{N} \sum_i \max_j n_{ij}$$

which measures the proportion of correctly assigned cells per predicted cluster.

These complementary metrics jointly evaluate the accuracy, separability, consistency, and compositional purity of the resulting cell clusters, providing a comprehensive assessment of clustering performance.

C Additional Experiments

C.1 t-SNE Results

The six t-SNE panels provide a qualitative comparison of cell embeddings produced by different encoder types on the integrated single-cell proteomics datasets of N2 and nanoPOTS. The GCN encoder shows moderate separation between some cell types, but clusters are diffuse and partially overlapping, with uneven batch mixing. The Beta Kernel Direct encoder yields compact, well-defined clusters with substantially improved batch mixing, suggesting strong alignment and biological signal preservation. The RWR-Truncated encoder produces scattered clusters with elongated or split shapes and inconsistent mixing, indicating an underrepresentation of the global topology. The RWR-Interpolated encoder improves cluster coherence and dataset alignment relative to the truncated version, better preserving cell identities. The Heat Kernel-Approximated encoder generates compact clusters with good local separation, though some overlap remains, while the Heat Kernel-Truncated encoder shows tight, distinct clusters with robust batch harmonization. Overall, the Beta Kernel Direct and Heat Kernel-Truncated encoders visually perform best, RWR-Interpolated and Heat Kernel-Approximated show moderate performance, and GCN and RWR-Truncated are the weakest. These results highlight that spectral- and kernel-based encoders generally produce more biologically meaningful embeddings than baseline GCNs or truncated proximity encoders.

C.2 Effect of Polynomial Degree

In Fig. 6, we evaluate the clustering performance of the interpolated Heat Kernel, interpolated RWR kernel, and the direct Beta Kernel across varying polynomial degrees K on the Scope2_Specht dataset. Across all four evaluation metrics—ARI, ASW, NMI, and PS—the interpolated Heat Kernel generally exhibits strong and stable behavior for moderate values of K, while the interpolated RWR kernel shows a gradual improvement, particularly at higher polynomial degrees. The direct Beta Kernel remains consistently competitive, achieving robust performance without requiring interpolation. When compared against the non-spectral baseline GCN Encoder, all three kernel-based methods demonstrate comparable or superior results, especially in terms of ASW and PS. Overall, the figure highlights the advantages of Arnoldi-orthonormalization-based spectral GCN approach, showing that increasing the polynomial degree can improve clustering stability and quality, particularly for the interpolated RWR-based representations.

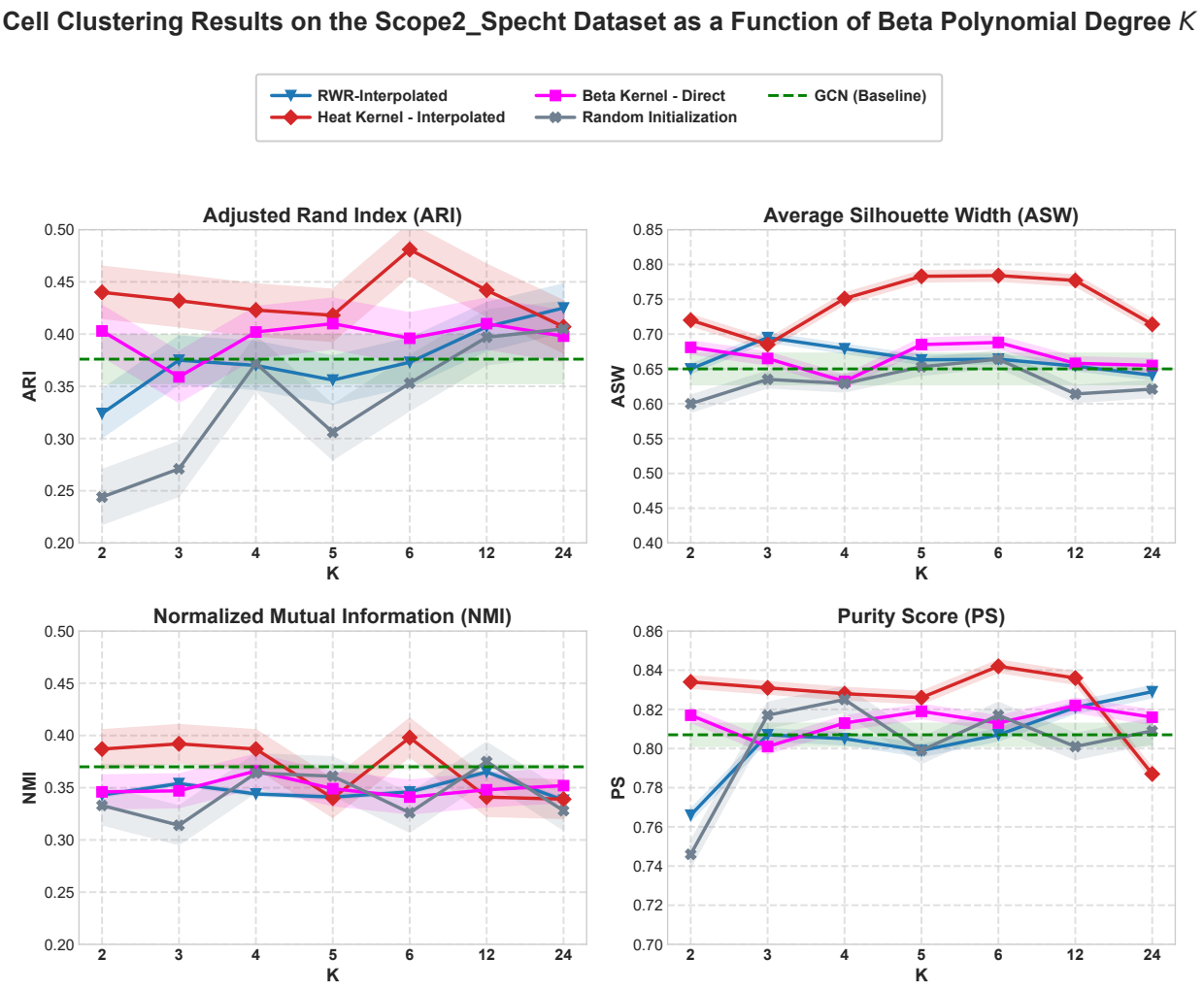

Fig. 6: Comparison of cell clustering performance on the `Scope2_Specht` dataset using Beta Kernel, Heat Kernel (Interpolated), and RWR-Interpolated filters across varying polynomial degrees $K \in \{2, 3, 4, 5, 6, 12, 24\}$. The four evaluation metrics include ARI,ASW, NMI, and PS. non-spectral GCN-Encoder is shown for baseline reference. The results illustrate how spectral interpolation and polynomial degree influence clustering performance across multiple criteria. The figure shows Mean ± 95% Confidence Interval over eight different seed runs.

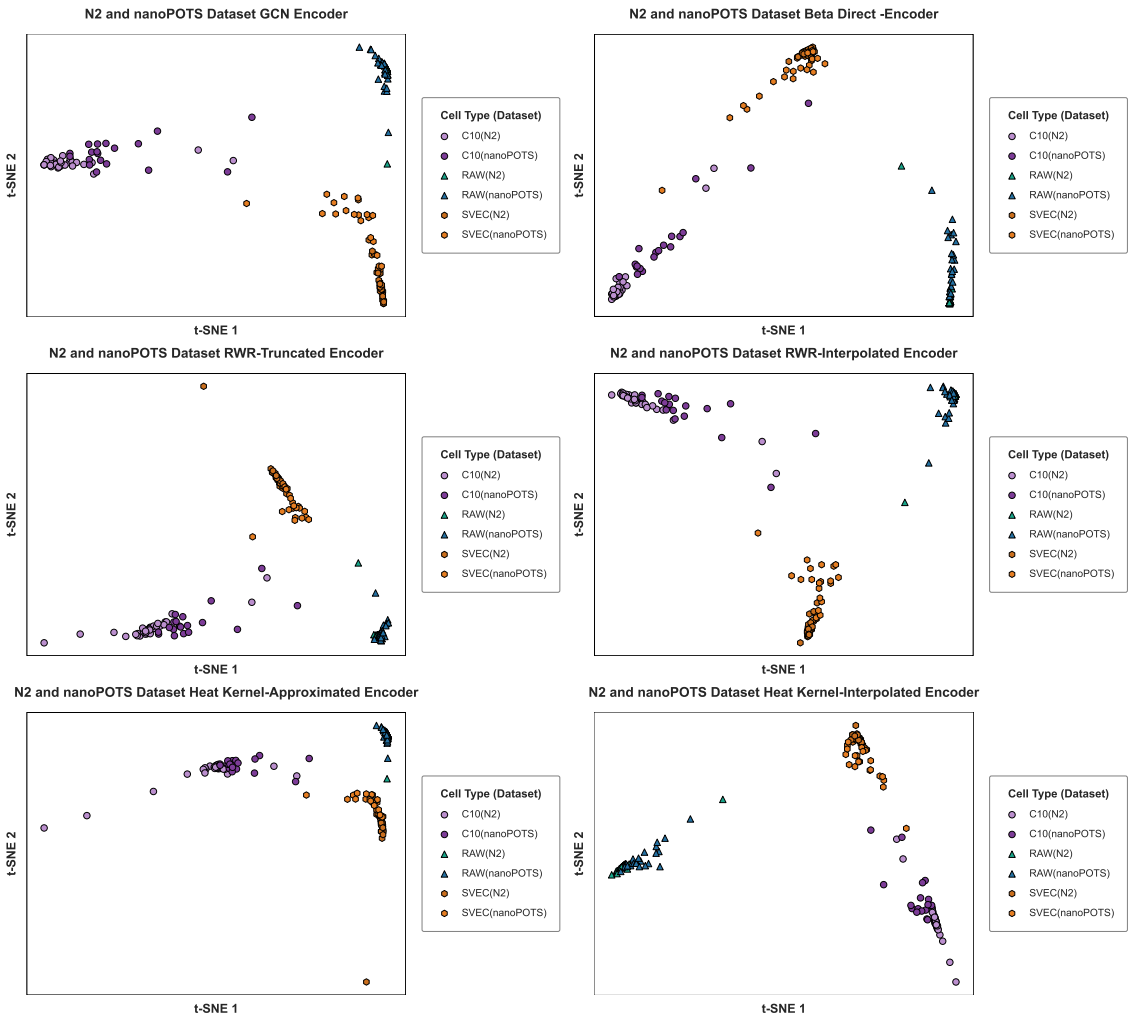

Fig. 7: t-SNE visualizations of cells from the N2 and nanoPOTS datasets. Each plot shows cells colored by their data acquisition batch and cell type. The six panels correspond to the different encoders: (i) GCN Encoder, (ii) Beta Kernel Direct, (iii) RWR Truncated, (iv) RWR Interpolated, (v) Heat Kernel Approximated, and (vi) Heat Kernel Truncated.

C.3 Effect of Embedding Dimension

To assess how the embedding dimension D influences clustering performance, we evaluated multiple metrics—Adjusted Rand Index (ARI), Average Silhouette Width (ASW), Normalized Mutual Information (NMI), and Purity Score (PS)—across six encoder types on the Scope2_Specht dataset (Figure 8).

Overall, moderate dimensions ($D = 64$) tend to yield the best performance for most encoders. The GCN encoder achieves its highest ARI and NMI at $D = 64$, while larger dimensions do not consistently improve clustering quality and sometimes reduce cluster coherence. Beta Kernel Direct embeddings exhibit relatively stable performance across dimensions, with peak ARI at $D = 64$ and minimal variation in ASW and PS. Heat Kernel Interpolated embeddings also favor $D = 64$, showing improved ARI, ASW, and NMI, whereas the Heat Kernel Approximated encoder demonstrates slightly declining performance as D increases beyond 64. For RWR-based encoders, the RWR-Interpolated method performs best at moderate dimensions, while RWR-Truncated shows a delayed peak at $D = 256$, reflecting sensitivity to global topological information.

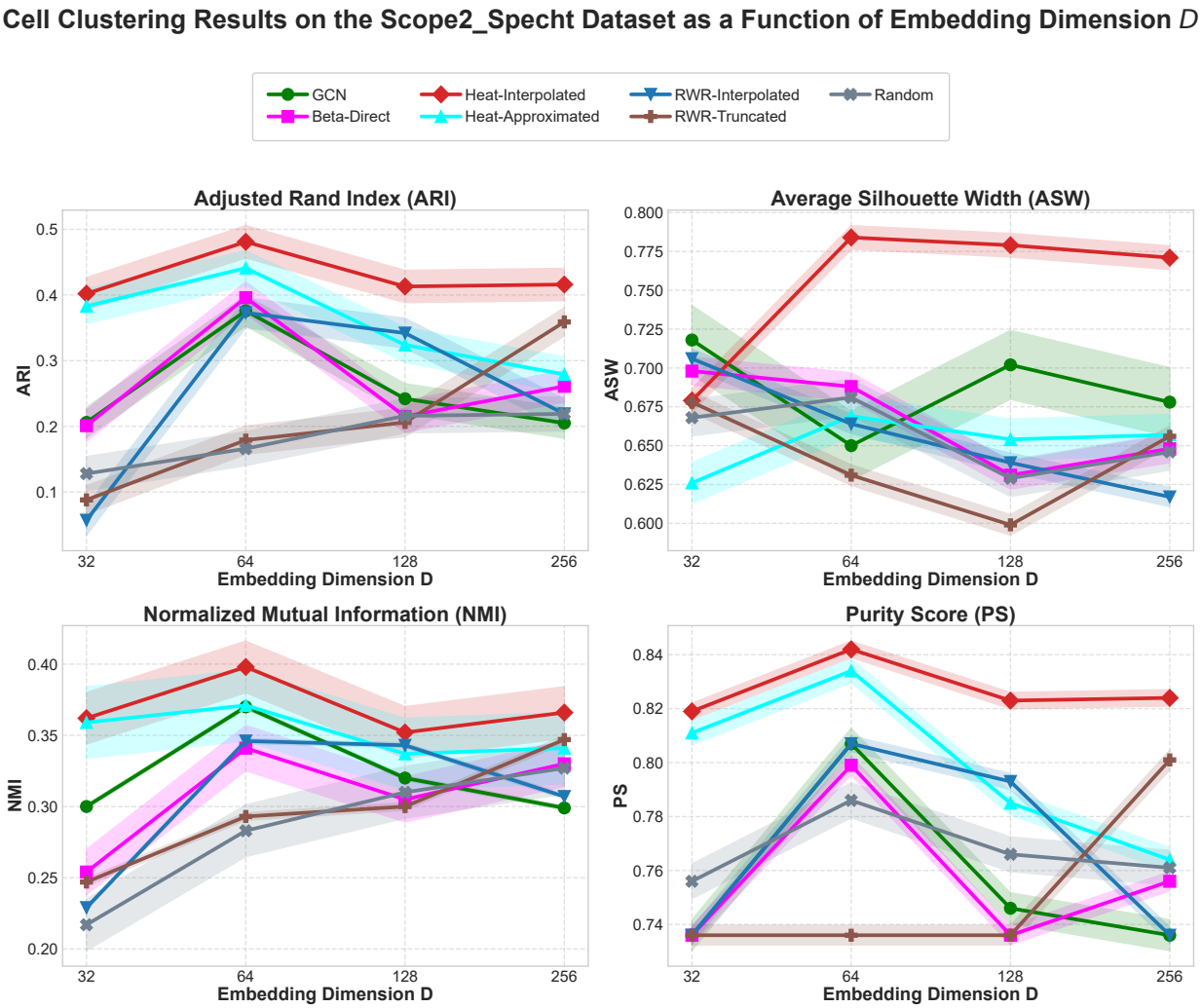

Fig. 8: Clustering performance as a function of the embedding dimension $D \in \{32, 64, 128, 256\}$ for six graph-based representation learning methods. The evaluation includes ARI, ASW, NMI, and PS metrics, with the x-axis shown on a logarithmic scale to reflect the exponential growth of D. The figure shows Mean ± 95% Confidence Interval over eight different seed runs.

Table 2: Summary of hyperparameters used in the experiments.

Hyperparameter	Value	Description
`stage1`	True	Start from Stage 1 preprocessing
Learning rate	1×10^{-3}	Optimization step size
Hidden dimension	64	GNN latent dimension
Projection head dim.	256	Dimension of projection layer
Activation	ReLU	Nonlinear activation function
# GCN layers	3	Depth of encoder network
# Prototypes	2	Structural prototype count
Perturbed edges	50	Added/removed edges for augmentation
Topology denoising	False	Enable structural denoising
DropEdge rate (view 1)	0.2	Edge dropout for view 1
DropEdge rate (view 2)	0.4	Edge dropout for view 2
Feature mask (view 1)	0.4	Feature dropout for view 1
Feature mask (view 2)	0.2	Feature dropout for view 2
α	0.05	Weighting factor in contrastive loss
τ	0.4	Temperature parameter
Weight decay	1×10^{-5}	L_2 regularization
Epochs	200	Number of training epochs
Random seed	39788	Reproducibility seed
Graph threshold	0.15	Similarity cutoff for edges
Feature preprocessing	True	Apply feature normalization

C.4 Hyperparameters used in the experiments

To ensure stable and reproducible training, we follow a consistent set of hyperparameters across all experiments. The framework begins from `Stage 1` by default and applies feature preprocessing before graph construction using a similarity threshold of 0.15. We train the encoder for 200 epochs with a learning rate of 10^{-3}, weight decay of 10^{-5}, and a random seed of 39788. The GNN backbone uses a hidden dimension of 64, six propagation layers, and ReLU activations. For contrastive learning, we adopt a projection head of dimension 256, two structural prototypes, and a balance coefficient $\alpha = 0.05$. Data augmentations include edge dropping (0.2/0.4 for the two graph views) and feature masking (0.4/0.2). Additional parameters include a temperature $\tau = 0.4$, a perturbation budget of 50 edges, and an optional topology-denoising module, disabled by default. All experiments are performed on GPU (`cuda:0`) when available.

C.5 Effect of Graph Edge Cut Threshold

Table 3: Edge sparsity of the Scope2_Specht dataset under different cutoff thresholds h.

Threshold (h)	0.05	0.15	0.30	0.50	0.75
Number of Edges	451,376	274,816	97,592	10,244	889

To study the influence of the graph edge cut threshold h on clustering performance, we evaluate multiple metrics—Adjusted Rand Index (ARI), Average Silhouette Width (ASW), Normalized Mutual Information (NMI), and Purity Score (PS)—across different embedding and propagation methods on the Scope2_Specht dataset (Figure 9). In our graph construction procedure, we first compute a cellwise similarity matrix S using the Pearson correlation coefficient (PCC) between the abundance feature vectors of cells, i.e.,

$$S_{ij} = \mathrm{PCC}(v_i, v_j) = \frac{\mathrm{cov}(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}},$$

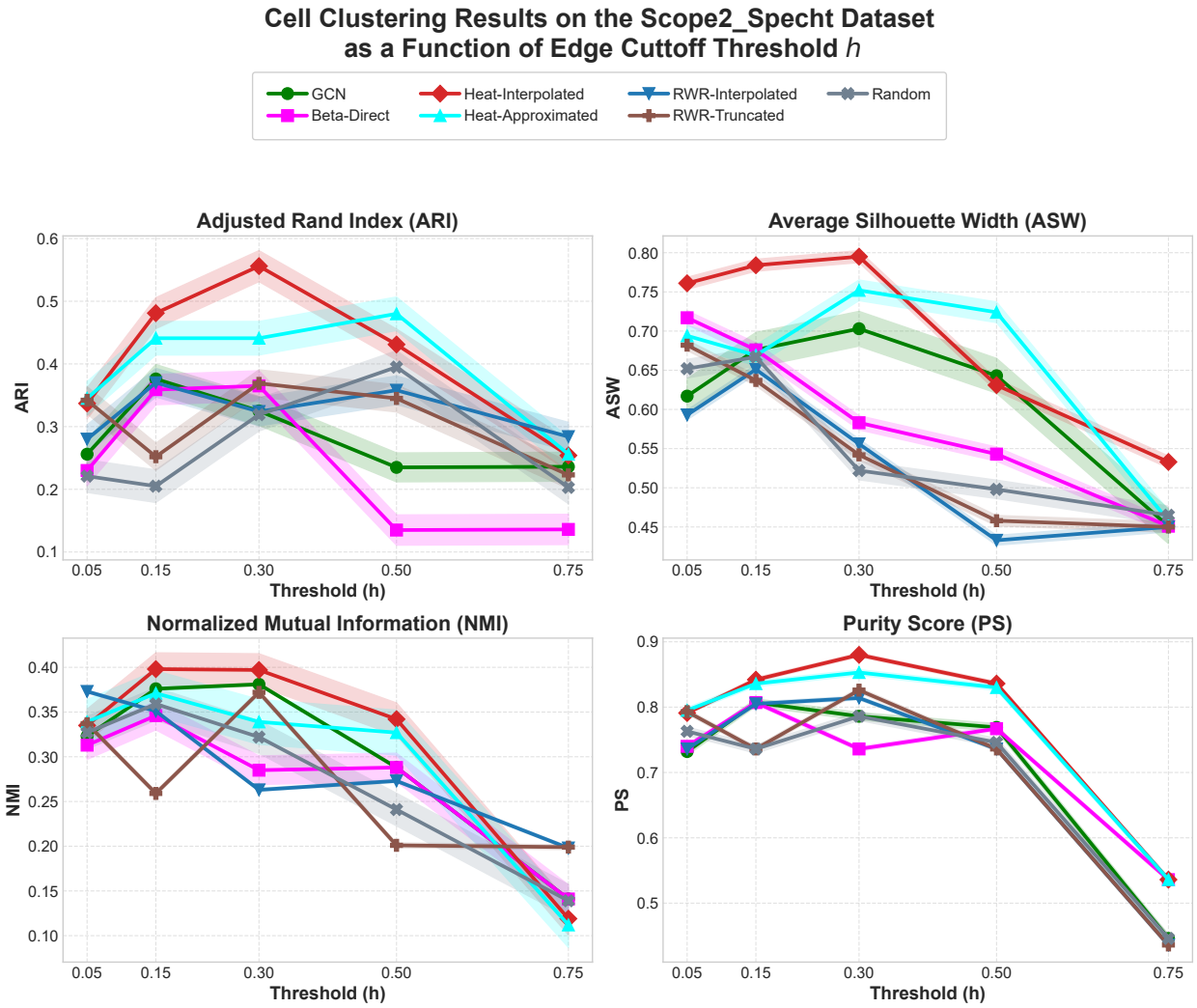

Fig. 9: Clustering performance as a function of the Graph threshold,similarity cutoff for edges $h \in \{0.05, 0.15, 0.3, 0.5, 0.75\}$ for seven graph-based representation learning methods. The evaluation includes ARI, ASW, NMI, and PS metrics, with the x-axis shown on a growth of h, increasing the sparsity. The figure shows Mean ± 95% Confidence Interval over eight different seed runs.

where $\mathrm{cov}(\cdot,\cdot)$ and σ denote the covariance and standard deviation, respectively. Then, a threshold h is applied to obtain the adjacency matrix $A \in \mathbb{R}^{N\times N}$ of the cell graph, defined as

$$A_{ij} = \begin{cases} 1, & \text{if } S_{ij} > h, \\ 0, & \text{otherwise.} \end{cases}$$

In this setting, the edge cut threshold h directly controls the sparsity of the constructed graph: smaller values of h yield denser graphs, while larger values result in sparser topologies.

Overall, moderate edge cut values (e.g., $h \in [0.2, 0.3]$) lead to the most stable and consistently strong performance across all evaluation metrics. When h is too small (e.g., $h = 0.1$), the resulting graph becomes overly dense, introducing noisy or weakly informative edges that blur cluster boundaries and slightly degrade ARI and ASW. In contrast, overly large thresholds (e.g., $h \geq 0.4$) produce sparse graphs that may fragment the underlying data manifold, which negatively impacts NMI and Purity for several methods.

Among the compared approaches, GCN-based embeddings exhibit clear performance peaks at moderate thresholds, highlighting their reliance on a well-balanced graph topology. Beta Kernel and Heat Kernel variants remain relatively robust across different values of h, but still achieve their best or near-best results in the moderate range, indicating that an appropriate edge cut improves both cluster separability and coherence. RWR-based methods are more sensitive to the choice of h, with noticeable performance drops at extreme thresholds, reflecting their dependence on meaningful global connectivity.

These results demonstrate that the edge cut threshold h is a critical hyperparameter in cell graph construction. Selecting a moderate threshold effectively balances noise suppression and structural preservation, leading to more discriminative and stable embeddings for downstream clustering.

C.6 Runtime Analysis

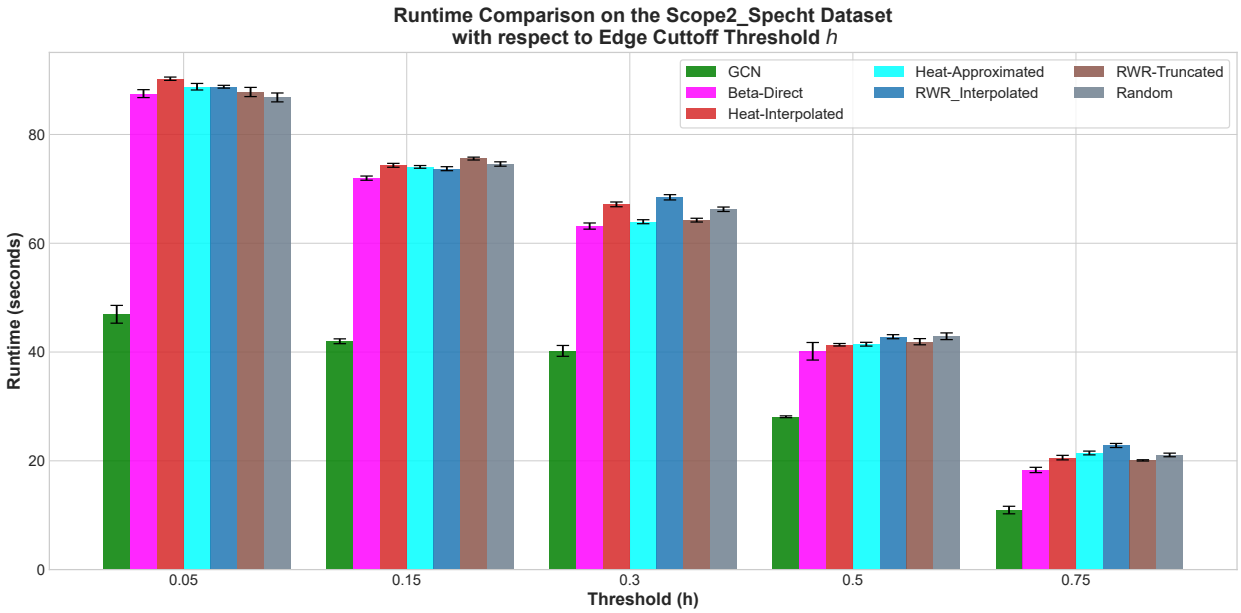

Fig. 10: Runtime comparison of different methods across threshold values (h). The bars represent the average runtime over 8 independent runs, and the error bars indicate the 95% confidence intervals. Results show how computational cost varies with the edge cutoff threshold for each method on the Scope2_Specht dataset.

The runtime analysis in Figure 10 shows clear computational differences among the methods. GCN-based approaches are consistently the fastest due to their localized message-passing mechanism and reliance on sparse matrix operations,

which scale efficiently with the number of edges. In contrast, Heat- and RWR-based methods incur higher computational cost because they involve diffusion processes, iterative propagation, and polynomial or spectral approximations that require additional matrix computations. Moreover, the observed decrease in runtime as the threshold h increases can be attributed to the induced sparsity of the graph, which reduces the number of active edges and thus the overall computational burden. The small confidence intervals further indicate stable and consistent runtime behavior across runs.