Simple GNNs with Low Rank Non-parametric Aggregators

Luciano Vinas and **Arash A. Amini**
University of California, Los Angeles
lucianovinas@g.ucla.edu

Abstract

We revisit recent spectral GNN approaches to semi-supervised node classification (SSNC). We posit that state-of-the-art (SOTA) GNN architectures may be over-engineered for common SSNC benchmark datasets (citation networks, page-page networks, etc.). By replacing feature aggregation with a non-parametric learner we are able to streamline the GNN design process and avoid many of the engineering complexities associated with SOTA hyperparameter selection (GNN depth, non-linearity choice, feature dropout probability, etc.). Our empirical experiments suggest conventional methods such as non-parametric regression are well suited for semi-supervised learning on sparse, directed networks and a variety of other graph types commonly found in SSNC benchmarks. Additionally, we bring attention to recent changes in evaluation conventions for SSNC benchmarking and how this may have partially contributed to rising performances over time.

1 Introduction

The problem of semi-supervised node classification (SSNC) [1, 2] has been a focal point in graph-based classification for roughly 20 years. At the task's inception, classical methods such as label propagation [3] and kernel learning [2] had seen moderate success in predicting unobseved node labels. Now, in an era where computation is more plentiful, modern approaches to the classification problem on graphs make use of the multilayer Graph Neural Network (GNNs) [4]. These networks, trained to predict node labels in SSNC, draw on both the individual node features ($\boldsymbol{X}$) and the broader network structure ($\boldsymbol{A}$) to inform their prediction.

The fundamental premise of SSNC is that the network structure allows us to borrow information from neighboring nodes for which we lack a response. This borrowing can enhance the prediction of the unobserved responses ($\boldsymbol{y}$) beyond what could be achieved with a traditional regression solely on node features. Recently, there has been a wide breadth of literature [5–7] which attempts to better leverage the network structure of the graph using GNNs. This recent flurry of activity has led to the proposal of many competing, and often intricate, architectures to solve the SSNC problem.

Our study of the leading GNN architectures and the benchmarks used to prove their algorithmic effectiveness, has led us to believe that many of the design choices found in modern GNNs may be drastically simplified, or even removed completely, at little-to-no cost to predictive performance. In our efforts to validate model performances, we revisit traditional estimation techniques like non-parametric regression. These techniques happen to be very effective for SSNC and highlight the importance of learnable feature aggregation in SSNC problems.

To this end, we devise a flexible non-parametric learner for feature aggregation. This learner generalizes the specific polynomial form used in spectral GNNs [8, 9]. That is, given a singular value decomposition of the network graph $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, the non-parametric learner $f : \mathbb{R} \to \mathbb{R}$ transforms the spectrum of $\boldsymbol{A}$ to produce a new aggregation matrix

$$\boldsymbol{P}_f = \boldsymbol{U} f(\boldsymbol{\Sigma}) \boldsymbol{V}^T$$

where f is applied entry-wise across the diagonal of $\mathbf{\Sigma}$. This singular value extension to the previous symmetric spectral approach of [9] helps clear a directed graph hurdle faced by previous spectral GNN techniques.

Our contributions are as follows:

1. Propose a nonparametric approach to learn f, hence a GNN aggregation operator, by borrowing ideas from the theory of reproducing kernel Hilbert spaces (RKHS), thus generalizing polynomial aggregation to a much broader class of spectral functions. By controlling the underlying kernel, one can impose different regularity constraints on the spectral filters.
2. Highlight the importance and sensitivity of nonparametric spectral reshaping and show how it can be used to simpify model hyperparameters (e.g. dropout probabilities, model depth, parameter-specific optimizers) at near-no-cost to SOTA performance.
3. Classification improvements of +5% and +20% compared to competing spectral methods and other non-linear GNN baselines for the challenging benchmark datasets Chameleon and Squirrel [10].
4. Outline common evaluation practices which have an outsized effect on model performance.

By standardizing evaluation practices and simplifying modeling considerations, we aim to disambiguate performance in the GNN model-space and hope to encourage more interpretable models and heuristics for future SSNC problems.

2 GNN and SSNC Formalism

In our observation framework, we consider observing a, potentially noisy realization, of the network with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. Specifically, each node in the network $i \in [n]$ is associated with a feature vector $\boldsymbol{x}_i$ and a label $y_i \in [C] := \{1, \ldots, C\}$.

In SSNC, it is assumed that for a subset of nodes $\mathcal{O} \subset [n]$ the labels $(y_i)_{i\in\mathcal{O}}$ are observed. In this setting, both the adjacency matrix $\boldsymbol{A}$ and the feature matrix $\boldsymbol{X}$ are assumed to be fully observed. The goal then is to correctly predict unobserved labels $(y_i)_{i\in\mathcal{O}^c}$ from the previously stated knowns.

GNNs are designed layerwise, with non-linearity $\phi^\ell : \mathbb{R} \to \mathbb{R}$, weight matrix $\boldsymbol{W}^\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and aggregation matrix $\boldsymbol{P}^\ell \in \mathbb{R}^{n\times n}$ all depending on layer $\ell \in [L]$. Placed altogether, the intermediate features of the GNN can be expressed as

$$\boldsymbol{Z}^{\ell+1} = \phi^\ell(\boldsymbol{P}^\ell \boldsymbol{Z}^\ell \boldsymbol{W}^\ell) \tag{1}$$

with ϕ^ℓ applied element-wise, $d_0 = d$ and $\boldsymbol{Z}^1 = \boldsymbol{X}$. In the case of a C-class classification problem, it is common to extract row-wise "argmax"s of the final features $\boldsymbol{Z}^L \in \mathbb{R}^{n\times C}$ using differentiable argmax surrogates such as $\mathrm{softmax}$. Choice of the aggregation matrix $\boldsymbol{P}^\ell$ may vary dramatically depending on architecture, but common choices include the adjacency matrix $\boldsymbol{A}$, its transformed variants (e.g. normalized Laplacian), and other, learnable, attention-based mechanisms [5].

2.1 Nonparametric Spectral Reshaping

In our proposed model, we consider the simplest variant of GNN: a one layer ($L = 1$), linear GNN, that is $\phi = \mathrm{id}$, where special attention is paid to the propagation structure $\boldsymbol{P}$. For ease of exposition, we first consider the undirected case where the adjacency matrix $\boldsymbol{A}$ is symmetric. Let $\boldsymbol{M}$ be a (symmetric) *network matrix* derived from $\boldsymbol{A}$. Examples include $\boldsymbol{M} \in \{\boldsymbol{A}, \boldsymbol{D}-\boldsymbol{A}, \hat{\boldsymbol{A}}, \boldsymbol{I}-\hat{\boldsymbol{A}}\}$ where $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. Our approach is to consider a general nonlinear deformation of $\boldsymbol{M}$, namely, $f(\boldsymbol{M})$ where $f : \mathbb{R} \to \mathbb{R}$ is a univariate function extended to the space of symmetric matrices by the so-called *functional calculus*. More precisely, given the eigendecomposition $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ of the $\boldsymbol{M}$ matrix, where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_i, i \in [n])$, one has

$$f(\boldsymbol{M}) = \boldsymbol{U} f(\boldsymbol{\Lambda}) \boldsymbol{U}^T$$

where $f(\boldsymbol{\Lambda}) = \mathrm{diag}(f(\lambda_i), i \in [n])$ is the natural extension of f to diagonal matrices. This way of extending univariate functions to self-adjoint operators has a long history in operator theory. Thus, our propagation operator is $\boldsymbol{P}_f = f(\boldsymbol{M})$ and we propose to optimize a loss over a general class of functions $\mathcal{F}$:

$$\hat{f} = \underset{f\in\mathcal{F},\, \boldsymbol{W}\in\mathbb{R}^{d\times C}}{\arg\min} \sum_{i\in\mathcal{O}} \ell\big(y_i, (f(\boldsymbol{M})\boldsymbol{X}\boldsymbol{W})_i\big) + \text{pen}(f) \tag{2}$$

where $\text{pen}(f)$ is some regularization penalty on f. Our main claim is that rather than assuming a specific parametric form for f, one can allow f to range in a potentially infinite-dimensional function space $\mathcal{F}$.

Of particular interest to us is when $\mathcal{F} = \mathbb{H}$, a reproducing kernel Hilbert space (RKHS) of functions, characterized by a kernel function $\mathcal{K} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. In such a space, the Hilbert norm $\|f\|_{\mathbb{H}}$ measures irregularity of f. Then, as long as $\text{pen}(f)$ is a monotonic function of the Hilbert norm $\|f\|_{\mathbb{H}}$, by the so-called represented theorem [11], problem (2) reduces to

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n,\, \boldsymbol{W}}{\arg\min} \sum_{i\in\mathcal{O}} \ell\big(y_i, (\boldsymbol{P}_{\mathcal{K}}(\boldsymbol{\alpha})\boldsymbol{X}\boldsymbol{W})_i\big) + \widetilde{\text{pen}}(\boldsymbol{\alpha}), \tag{3}$$

$$\text{where} \quad \boldsymbol{P}_{\mathcal{K}}(\boldsymbol{\alpha}) := \boldsymbol{U}(\text{diag}(\boldsymbol{K}\boldsymbol{\alpha}))\boldsymbol{U}^T, \tag{4}$$

and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is the kernel matrix with entries $K_{ij} = \mathcal{K}(\lambda_i, \lambda_j)$. If $\text{pen}(f) = \omega(\|f\|_{\mathbb{H}})$ for monotonic function $\omega : \mathbb{R}_+ \to \mathbb{R}$, then $\widetilde{\text{pen}}(\boldsymbol{\alpha}) = \omega(\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha})$. Given $\hat{\boldsymbol{\alpha}}$ one can explicitly write down the solution $\hat{f}$ of the functional problem (2) as

$$\hat{f}(\lambda) := \sum_j \hat{\boldsymbol{\alpha}}_j \mathcal{K}(\lambda, \lambda_j)$$

which is the learned spectral filter.

Practical considerations. We found slight improvements in performance when regularizing with $\boldsymbol{\alpha}^T\boldsymbol{\alpha}$ rather than the Hilbert norm surrogate ($\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}$).This amounts to using $\widetilde{\text{pen}}(\boldsymbol{\alpha}) = \rho\,\boldsymbol{\alpha}^T\boldsymbol{\alpha}$ for some $\rho > 0$. When minimizing with GD type methods, this is equivalent to introducing weight decay ρ, and is already built into SOTA solvers.

Additionally, we consider the possibility that edges in the network themselves have a component of randomness associated with them (Section 2.2). This means that our initial spectral inputs $\lambda_1, \lambda_2, \ldots, \lambda_n$ are themselves noisy. It is then natural to truncate the spectral decomposition of $\boldsymbol{M}$ to the top r eigenvalues (in absolute values). Thus, if the eigenvalues are ordered as $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$, we consider $\boldsymbol{M}^{(r)} = \boldsymbol{U}\boldsymbol{\Lambda}^{(r)}\boldsymbol{U}^T$ where $\boldsymbol{\Lambda}^{(r)} = (\lambda_i, i \in [r])$ and let the aggregation matrix be $f(\boldsymbol{M}^{(r)}) = \boldsymbol{U}f(\boldsymbol{\Lambda}^{(r)})\boldsymbol{U}^T$. Following through as before, the only changes to the algorithm is to replace $\boldsymbol{K}$ in (4) with $\boldsymbol{K}^{(r)} = (\mathcal{K}(\lambda_i, \lambda_j))_{i,j=1}^r$. We also note that $\boldsymbol{\alpha}$, the learnable spectral parameter, will be r-dimensional in this case. We treat the $r \in [n]$ as a hyperparameter and study its effect in simulations. We refer to the case $r < n$ as low-rank (LR) kernel model.

Directed/asymmetric case. All the above naturally extends to directed networks, where $\boldsymbol{M}$ is not necessarily symmetric, by replacing the eigenvalue decomposition with the SVD: $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ where $\boldsymbol{\Sigma} = \text{diag}(\sigma_i, i \in [n])$ collects the singular values of $\boldsymbol{M}$. The aggregation matrix in this case is $\boldsymbol{P}_f = \boldsymbol{U}f(\boldsymbol{\Sigma})\boldsymbol{V}^T$ and its finite-dimensional version is $\boldsymbol{P}_{\mathcal{K}}(\boldsymbol{\alpha}) = \boldsymbol{U}(\text{diag}(\boldsymbol{K}\boldsymbol{\alpha}))\boldsymbol{V}^T$ with the kernel matrix $\boldsymbol{K} = (\mathcal{K}(\sigma_i, \sigma_j))_{i,j=1}^n$ now based on singular values. Everything else follows similarly, including rank truncation, where we use ordered singular values instead.

Multiple layers. We mainly focus on a single-layer model (with identity activation) and empirically show that a single layer of this model is enough to achieve near SOTA performance. However, it is straightforward to extend the model to multiple layers via the general blueprint (1) where each layer will have aggregation operator $\boldsymbol{P}^\ell = f_\ell(\boldsymbol{M})$ with f_ℓ belonging to $\mathbb{H}$.

2.2 Motivating General Spectral Learners

Implicit in all graph learning problems is the assumption that node features $\boldsymbol{X}$ are only partially informative towards predicting $\boldsymbol{y}$. To motivate why a spectral GNN of the form (2), with a general reshaping function f can improve prediction, let us consider perhaps the simplest theoretical model of SSNC, the so-called Contextual Stochastic Block Model (CSBM) [12]. The idea is that the labels $\boldsymbol{y}$ are latent variables generating both $\boldsymbol{A}$, via a C-class SBM: $\mathbb{P}(A_{ij} = 1 \,|\, \boldsymbol{y}) = B_{y_i,y_j}$, and the node features via a mixture model: $\boldsymbol{x}_i \,|\, y_i \sim N(\boldsymbol{\mu}_{y_i}, \sigma^2 I)$.

In this case, the idealized version of $\boldsymbol{A}$ is $\mathbb{E}[\boldsymbol{A}]$ which is a rank C matrix with C eigenvectors that are *indicator vectors* of each of the C classes $\Gamma_1, \ldots, \Gamma_C \in \{0,1\}^n$. Consequently, if $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ is the EVD of $\boldsymbol{A}$, one expects $\boldsymbol{U}_{:j} \approx \mathbf{1}_{\Gamma_j}$ for $j \in [C]$, while $\boldsymbol{U}_{:j}$ for $j > C$ are expected to be mostly noise. So an ideal aggregation operator is close to $f(\boldsymbol{M}) = \boldsymbol{U}f(\boldsymbol{\Lambda})\boldsymbol{U}^T$ where f is a step function that passes the $\lambda_j, j \in [C]$ through and zero out the rest. As the experiment in Section 3.2 show, this is mostly what happens when we train (2) on CSBM, albeit with more nuance. In finite samples, $\boldsymbol{A}$ is not exactly low-rank and lower eigenvectors might still have information about $\boldsymbol{y}$. This is what we observe in practice where the learned f is a *tapered* thresholding operator, that gradually downweights lower frequencies.

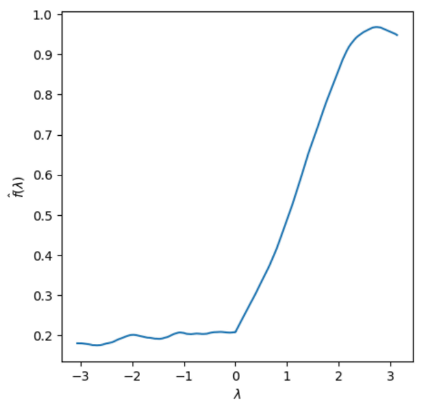

Figure 1: Example of learned spectral filter $\hat{f}(\lambda)$ for the CSBM experiment with an unbounded Sobolev kernel ($\gamma = 0.1$).

The general low-rank behavior of the $\mathbb{E}[\boldsymbol{A}]$ is not limited to SBMs and holds for more realistic network models such as random dot product graph (RDPG) [13] where depending on the distribution of latent positions, more complex tapering might be optimal.

2.3 Complexity of Low Rank Spectral Learners

Scalability remains an issue for dense spectral methods. However in the case of low-rank non-parametric aggregators, this issue can be addressed through the use of a low-rank spectral approximation. By first selecting a rank parameter r for the non-parametric aggregator, computation can be better budgeted ahead of time through for a graph $G = (V, E)$ through the use of a low-rank SVD approximations. Specifically for PyTorch, a low-rank, random SVD routine based on [1] is implemented in the function `torch.svd_lowrank`.

Computation and error complexity for this routine can be found in section 6.2 of [1]. For a sparse adjacency matrix given by G and a number of total iterations q, this routine has a time complexity of $\mathcal{O}(qr|E| + r^2|V|)$ and an error complexity, in operator norm, of $(r|V|)^{1/2(2q+1)}\sigma_{r+1}$. In total, we obtain a decomposition procedure which is: exponentially exact with respect to q, at most quadratic in time with respect to r, at most linear in time with respect to graph parameters $|V|$ and $|E|$.

In the forward pass of the non-parametric aggregator, a graph with d-dimensional node features and c classes will contribute a computational complexity of $\mathcal{O}(|V|c(d+r))$. Additionally, since the non-parametric aggregator is linear with respect to weights W and parameter α, gradient computation in the backward pass can re-use intermediaries found in the forward pass, potentially saving computation.

3 Experiments

In an effort to show the power of feature aggregation for SSNC problems, our modeling effects will focus entirely on the aggregation matrix $\boldsymbol{P}$. No modifications are made to the original features $\boldsymbol{X}$ or the structure of the linear weight $\boldsymbol{W}$. As such, in our experiments we do not consider any model-specific augmentations such as dropout [14], batchnorm [15], or per-parameter optimizers (i.e. different learning rates for different layers). The design of $\boldsymbol{P}$ will have the following degrees of freedom:

- **Matrix representation of network ($\boldsymbol{M}$):** We refer to any matrix $\boldsymbol{M}$ derived from algebraic manipulations of the adjacency of a network $\boldsymbol{A}$, to be a *matrix representation* of the network. In particular we consider the following two representations:
  - *Adjacency*: This is simply an identity transformation on $\boldsymbol{A}$ with $\boldsymbol{M} = \boldsymbol{A}$.
  - *Laplacian*: This is $\boldsymbol{M} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is the row-sum degree matrix of $\boldsymbol{A}$.
- **Spectral truncation factor (r):** Given a truncation factor r, the spectral system $(\boldsymbol{U}, \boldsymbol{\Lambda})$, resp. $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^T)$, will be reduced to $(\boldsymbol{U}_{:r}, \boldsymbol{\Lambda}_{:r})$, resp. $(\boldsymbol{U}_{:r}, \boldsymbol{\Sigma}_{:r}, (\boldsymbol{V}_{:r})^T)$, where the eigenvectors associated with the bottom $n - r$ eigenvalue magnitudes are dropped. In our experiments, spectral truncations from 0 to 95% in 5% intervals are considered.

	Cora	CiteSeer	PubMed	Chameleon	Squirrel	Actor	Cornell	Texas	Wisconsin
MLP2	77.8 ± 1.6	77.2 ± 1.1	88.2 ± 0.5	48.5 ± 2.6	34.8 ± 1.4	40.3 ± 2.3	86.1 ± 3.0	91.7 ± 4.4	95.0 ± 2.6
LINEAR	78.9 ± 2.0	76.2 ± 1.2	85.8 ± 0.4	48.1 ± 3.2	34.9 ± 1.4	38.9 ± 1.2	84.9 ± 5.6	89.7 ± 3.8	95.0 ± 3.8
AGG. LINEAR	84.0 ± 2.0	73.9 ± 1.4	82.6 ± 0.5	79.0 ± 1.4	78.0 ± 1.1	32.4 ± 1.3	67.8 ± 8.7	86.8 ± 3.5	83.8 ± 3.2
KERNEL	88.6 ± 1.0	81.1 ± 1.0	89.4 ± 0.8	78.7 ± 1.1	76.0 ± 1.2	32.2 ± 1.8	83.3 ± 5.9	88.2 ± 2.6	92.1 ± 3.4
LR KERNEL	—	—	—	79.4 ± 1.4	76.8 ± 1.3	32.3 ± 1.7	—	—	—
GPRGNN*	79.5 ± 0.4	67.6 ± 0.4	85.1 ± 0.1	67.5 ± 0.4	49.9 ± 0.5	39.3 ± 0.3	91.4 ± 0.7	92.9 ± 0.6	NA
SGC/ASGC*	73.9 ± 2.5	70.2 ± 1.0	79.1 ± 1.0	72.3 ± 0.9	59.0 ± 1.0	36.5 ± 0.8	86.8 ± 3.6	86.2 ± 3.1	NA
JACOBICONV*	89.0 ± 0.5	80.8 ± 0.8	89.6 ± 0.4	74.2 ± 1.0	55.8 ± 0.6	40.7 ± 1.0	92.3 ± 2.8	92.8 ± 2.0	NA
ACMII-GCN	89.0 ± 0.7	81.8 ± 1.0	90.7 ± 0.5	68.4 ± 1.4	54.5 ± 2.1	41.8 ± 1.2	95.9 ± 1.8	95.1 ± 2.0	96.6 ± 2.4

Table 1: Performance: Mean test accuracy ± std. dev. over 10 data splits. Models include our own variations of "Linear" and "Aggregated Linear" GNNs, along with other state-of-the-art (SOTA) GNNs. Dashed entry in for LR KERNEL signifies validated choice is the same as the full-rank KERNEL. Performance is comparable between our simple GNNs and SOTA in some cases. Results for GPRGNN, SGC/ASGC. JACOBICONV and ACMII-GCN are cited from [6], [19], [9], and [7] respectively. Entries marked with '$*$' report 95% confidence intervals.

- **Choice of kernel ($\mathcal{K}$):** In ordering our RKHS we select among the following kernels:
  - *Identity:* $K_{ij} = 1\{i = j\}$
  - *Linear (Outer product):* $\mathcal{K}(\sigma_i, \sigma_j) = \sigma_i \sigma_j$
  - *Compact Sobolev*: $\mathcal{K}(\sigma_i, \sigma_j) = \min(\sigma_i, \sigma_j)$
  - *Unbounded Sobolev:* $\mathcal{K}(\sigma_i, \sigma_j) = \exp(\gamma|\sigma_i - \sigma_j|)$
  - *Gaussian Radial Basis:* $\mathcal{K}(\sigma_i, \sigma_j) = \exp(\gamma|\sigma_i - \sigma_j|^2)$

  Note, in the case of identity, the "kernel" does not generate a continuous RKHS. For the last two kernels, the bandwidth parameter $\gamma \in \mathbb{R}_+$ can be determined on validation.

Note that, the choice of matrix representation $\boldsymbol{M}$ matters here insofar that it determines the "modes" or partitions of the network with its left and right eigenvectors $(\boldsymbol{U}, \boldsymbol{V})$.

For our optimizer, we use the standard Adam optimizer [16] with weight decay. For simplicity, both parameter $\boldsymbol{\alpha}$ and weight matrix $\boldsymbol{W}$ share the same weight decay under Adam.

3.1 SSNC Benchmarks

Our methods are evaluated against common SSNC benchmarks. The Chameleon, Squirrel, and Actor benchmarks contain directed networks, while the other benchmarks contain undirected networks. More information on all benchmarks can be found in Pei et al. [17]. All values are recorded using the *balanced splits* defined in Chien et al. [6]. Section 4 provides a comprehensive analysis on the impact of splitting conventions. Although not covered in this paper, alternative benchmarks for simple spectral models can be found in Zhu and Koniusz [18].

The following linear and kernel models are considered for evaluation: LINEAR ($\boldsymbol{XW}$), AGGREGATED LINEAR ($\boldsymbol{MXW}$), KERNEL ($\boldsymbol{P}_{\mathcal{K}}\boldsymbol{XW}$), and LR KERNEL ($\boldsymbol{P}_{\mathcal{K},r}\boldsymbol{XW}$). Model hyperparameters such as learning rate, weight decay, the specific aggregator $\boldsymbol{P}$ will be determined for each dataset using the mean accuracies of the validation splits. For completeness, we have also implemented a non-linear baseline which learns using only feature information $\boldsymbol{X}$. This model is a simple two-layer ReLU multi-layer perceptron MLP2 ($\phi(\boldsymbol{XW}^1)\boldsymbol{W}^2$) with hidden layer size determined on validation.

Our models and their results compared to other current SOTA methods can be found in Table 1. We note that, for almost all of the larger graph benchmarks, our models perform within uncertainty or better compared to SOTA. In particular for directed graphs like Chameleon and Squirrel, we see gains in accuracy as high as 5% and 20% over other SOTA methods. A point of emphasis here is the relative simplicity of our models compared to the performance they attain. The absence of any post-model augmentations distinguishes our approach from the implementations of other competing SOTA spectral methods like JACOBICONV [9].

A point of difficulty where the performance gap persists, is where the node response $\boldsymbol{y}$ is overwhelming described by its node information $\boldsymbol{X}$. Graphs with this property (Actor, Cornell, Texas, and

	max params.	$n = 300$	$n = 600$	$n = 1200$	$n = 1500$
X-ONLY ORACLE	0	64.3 ± 3.9	66.2 ± 2.9	63.3 ± 2.7	64.6 ± 1.4
KERNEL	1512	75.0 ± 3.5	86.6 ± 3.3	94.5 ± 1.2	97.3 ± 0.8
ACMII-GCN	102623	75.3 ± 6.4	89.5 ± 3.1	96.0 ± 1.2	97.7 ± 0.8

Table 2: Simulation experiments on a three-class CSBM. Mean test accuracy and std. dev. of 10 runs are reported. X-ONLY ORACLE is the accuracy associated with oracle classification on solely X. Maximum parameter counts for the two methods are also summarized. Relevant average degree Δ_n for the simulations are $\Delta_{300} = 1.83$, $\Delta_{600} = 3.68$, $\Delta_{1200} = 7.58$, and $\Delta_{1500} = 9.44$.

Wisconsin) can be identified by the negative performance gap between LINEAR and AGGREGATED LINEAR as well as the SOTA-like performance of MLP2. Note that, even without using any graph information, MLP2 is able to achieve SOTA within uncertainty on almost all of the X-dominated, network datasets. Furthermore, for the cases of Cornell, Texas, and Wisconsin, there is a possibility of running into sample size issues for graph based methods. With the exception of Actor, these datasets are only 100-200 nodes large (less than 1/10 the size of the other network benchmarks).

3.2 CSBM Experiment

To illustrate the effectiveness of nonparametric spectral learners, we performed an experiment on simulated CSBM data. We consider $C = 3$ classes and node features in $\mathbb{R}^3$ ($\boldsymbol{X} \in \mathbb{R}^{n\times 3}$) generated using `make_blobs` function of `scikit-learn` package with cluster standard deviation of 10. This leads to a hard classification problem for an oracle that only knows X, with optimal Bayes accuracy of roughly 0.63 (for large n). The SBM component has connection probabilities $B_{kk} = 0.015$ and $B_{k\ell} = 0.02$ for $k \neq \ell$. We vary the number of nodes n over $300, 600, 1200$, and 1500.

Table 2 summarizes the results for our nonparametric learner (KERNEL) and ACMII-GCN as a competing SOTA. Also shown is the average degree of the resulting networks. As n increases the CSBM model becomes more informative, which is reflected in increased prediction accuracy. At the two ends of the SNR spectrum ($n = 300$ and $n = 1500$) the performance of the KERNEL GNN and ACMII-GCN are very close, while there is a slight advantage for ACMII-GCN in the middle ($n = 600$ and $n = 1200$), though the two methods are still comparable due to the overlap of the wide uncertainty ranges.

What, however, is noteworthy is the significant effect of the spectral shaping in GNN performance: the KERNEL GNN significantly improves the performance beyond the X-only oracle with very few parameters and at very low graph SNRs; for example, at $n = 300$, where the parameter count is 312 and the average degree is barely 2 (a very weak graph signal). The simplicity of the KERNEL GNN allows us to exactly quantify the effect of nonparametric spectral learning since this is the only operation performed outside of applying the learned linear weights $\boldsymbol{W}$.

3.3 Aggregation Ablation

To understand the impact of the degrees of freedom defined for the aggregation matrix in section 3, we conduct an ablation study on the three hyperparameters: matrix representation $\boldsymbol{M}$, truncation factor r, and the choice of kernel $\mathcal{K}$.

Matrix Representation ($\boldsymbol{M}$): For this experiment we keep spectral truncation fixed at 0% and choose the best kernel through validation splits. In other words, this experiment is conducted using the full-rank KERNEL model with a best validated kernel fit to each dataset. In the experiment, we explore affects of fixing either $\boldsymbol{M} = \boldsymbol{A}$ or $\boldsymbol{M} = \boldsymbol{D} - \boldsymbol{A}$.

Figure 2 shows the accuracy change across datasets when using a Laplacian matrix representation $\boldsymbol{D} - \boldsymbol{A}$ rather than an adjacency matrix representation $\boldsymbol{A}$. As shown by the figure, directed graphs such as Chameleon and Squirrel show large benefits when using the adjacency matrix representation. Otherwise there seems to be a slight but persistent benefit in using the Laplacian representation for undirected datasets.

Kernel	Cora	CiteSeer	PubMed	Chameleon	Squirrel	Actor	Cornell	Texas	Wisconsin
Identity	78.8 ± 2.7	72.6 ± 2.0	81.6 ± 0.9	69.7 ± 2.7	44.9 ± 2.9	28.6 ± 3.0	60.4 ± 8.1	76.2 ± 4.3	71.6 ± 5.7
Sob. Cmpct.	75.1 ± 1.9	73.0 ± 1.4	88.5 ± 0.4	41.4 ± 2.2	33.2 ± 1.1	$\mathbf{32.2 \pm 1.8}$	$\mathbf{83.3 \pm 5.9}$	88.6 ± 4.0	$\mathbf{92.1 \pm 3.4}$
Linear	81.1 ± 2.0	72.1 ± 1.8	82.3 ± 1.0	$\mathbf{78.7 \pm 1.2}$	$\mathbf{76.0 \pm 1.2}$	31.6 ± 0.9	66.5 ± 6.1	77.2 ± 8.0	81.3 ± 4.8
Sob. Unbnd.	88.8 ± 0.8	$\mathbf{81.1 \pm 1.0}$	89.2 ± 2.0	54.5 ± 6.4	68.8 ± 8.2	30.7 ± 1.0	80.6 ± 6.4	$\mathbf{88.2 \pm 2.6}$	90.4 ± 5.6
Gauss. RBF	$\mathbf{88.6 \pm 1.0}$	80.3 ± 1.9	$\mathbf{89.4 \pm 0.8}$	60.4 ± 8.4	71.3 ± 4.4	30.4 ± 1.3	79.4 ± 5.3	84.0 ± 4.5	85.8 ± 4.7

Table 3: Impact of the kernel choice on the performance of the full-rank KERNEL model. Bold entries correspond to the model selected by validation.

Choice of Kernel ($\mathcal{K}$): For this experiment we once again use the full-rank KERNEL model. This time the matrix representation $\boldsymbol{M}$ is chosen through validation and the choice of kernel is varied across datasets. Table 3 shows performance results for the various choices of kernels. In Table 3, we see a complicated dependence between kernel choice and the accuracy of node prediction. Although some results are within uncertainty, the dependence between kernel regularity and SSNC performance is not immediately clear. In the case of the Chameleon and Squirrel datasets, it is apparent that the wrong choice in kernel may lead to significant performance degradations (up to $\sim$30%).

Spectral Truncation Factor (r): For this experiment, both the matrix representation and the choice of kernel have been selected based on best validation with truncation factor r fixed for the extent of each sub-experiment. Figure 3 demonstrates the effect of truncation on performance and how it gradually degrades with the truncation percentage. The rate at which performance degrades seems dependent on the dataset, but most benchmarks retain $\sim$90% performance even after a 40% spectral truncation. In special cases like Squirrel and Chameleon, performance can be seen to increase at larger truncation values.

Alleviating Kernel Dependent Performance: For this experiment, we explore the affects of kernel choice for the LR KERNEL model. In particular, we focus on the performance impact of kernel choice for the directed dataset benchmarks. Rather than reporting mean accuracies, Figure 4 shows the full violin plot of test split performances for each kernel-dataset combination. We notice a *homogenization* of results, where the choice of kernel is negligible to the overall SSNC performance. We stress however that this solution is partial, as the same order of homogenization is not observed for the other undirected datasets. Identifying relevant graph statistics which may describe this homogenization discrepancy is something which is left to future work.

4 Changes in Evaluation Conventions

The convention of using citation networks [20] (Cora, Citeseer, Pubmed) in SSNC benchmarks was popularized by the graph embedding work of Yang et al. [21]. Yang et al. [21] defined the "sparse" train-test split of the citation datasets and their node masks were made publicly available. The sparse split fixed *20 nodes per class for training* and *1000 nodes total for testing*. These values were held constant across citation datasets, meaning larger networks likes Pubmed were left with a relatively low label rate of $\sim$5%.

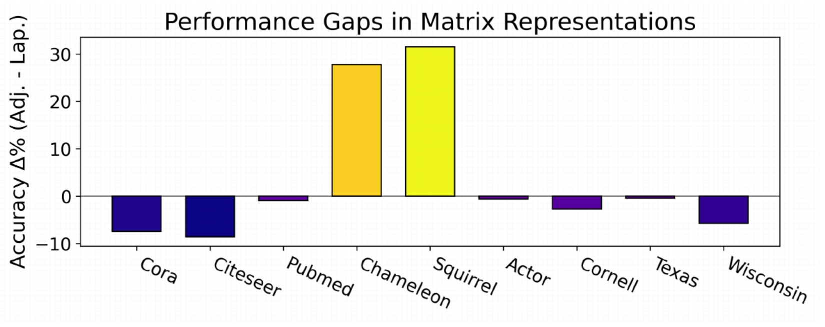

Figure 2: Accuracy comparison of the KERNEL model for different graph representations $\boldsymbol{A}$ and $\boldsymbol{D} - \boldsymbol{A}$. Shown above is the signed accuracy difference between the adjacency and Laplacian representations. Best performing kernel was selected per dataset.

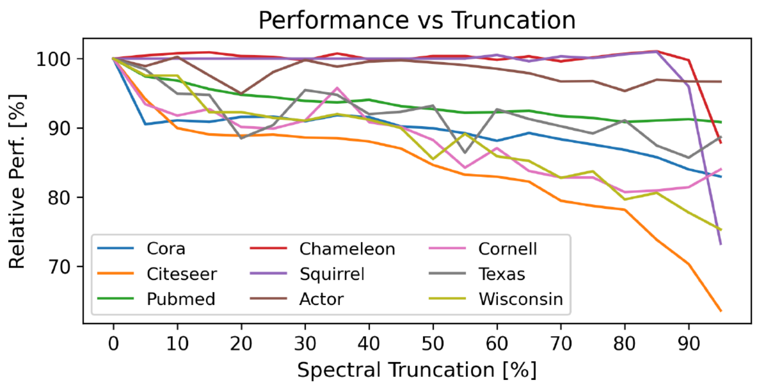

Figure 3: LR KERNEL performance relative to the full-rank KERNEL for different truncation factors r. Performance is seen to gradually decline on most datasets as the truncation factor r decreases (that is truncation percentage increases). LR KERNEL performance can also be seen to periodically increase above full-rank KERNEL performance for the datasets Chameleon (red) and Squirrel (purple).

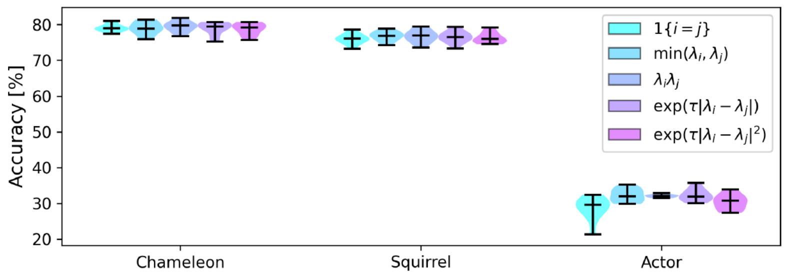

Figure 4: Performance homogenization achieved by LR KERNEL model on directed networks.

Quickly following was the semi-supervised work of Kipf and Welling [22] and Veličković et al. [5]. These follow-up papers defined a new “public" split where *500 previously unlabeled nodes* in the sparse split were now used for validation. In the respective code implementations of each paper, the additional labels were used for early stopping criteria and to determine the final model checkpoint.

Introduced later was the “dense” split by Pei et al. [17], where train, validation, and test were now fractions of the whole graph, set to *60%-20%-20% respectively*. This paper also popularized two new benchmark datasets, the WebKB dataset [23] (Cornell, Texas, Wisconsin) and the Wikipedia animal page-page networks [10] (Chameleon, Squirrel).

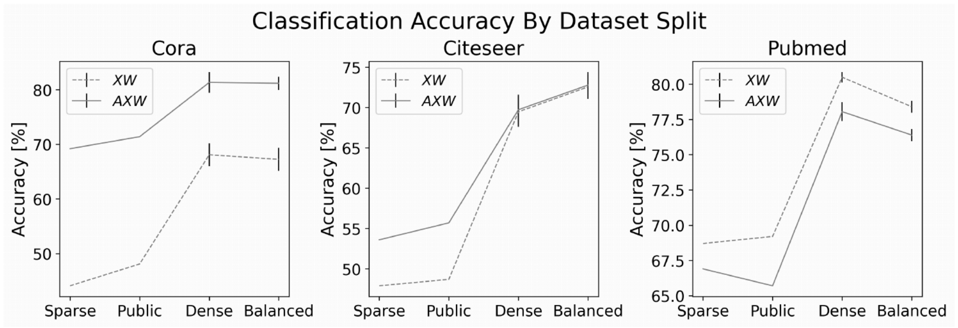

Figure 5: Accuracy results and uncertainties on the citation datasets using different splits with linear models $\boldsymbol{XW}$ and $\boldsymbol{AXW}$. “Public" refers to the split introduced by Kipf and Welling [22]. Both “Sparse" and “Public" are single splits, so one cannot associate uncertainty to them.

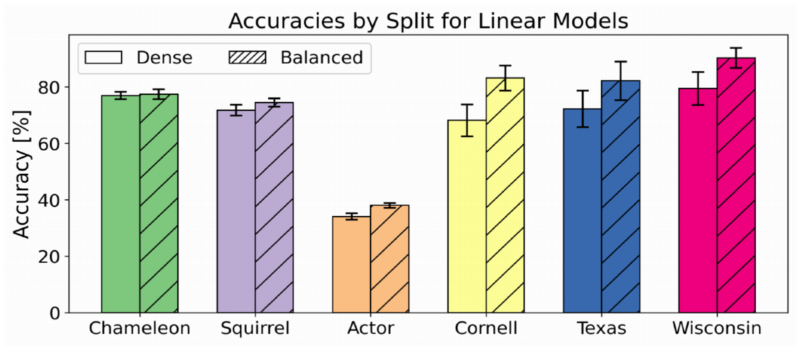

Figure 6: Accuracy results on datasets introduced by Pei et al. [17]. “Dense" refers to the original split while “Balanced" refers to the split introduced by Chien et al. [6]. Test results and uncertainties are evaluated using models $\boldsymbol{XW}$ and $\boldsymbol{AXW}$. Results shown are for method with best validation.

Most recently a “balanced" split was proposed by Chien et al. [6]. This is a class-balanced split where, *for each class in a network*, a 60%-20%-20% mask is made with then each class mask being collected into a final, aggregate train-validation-test split. Both the balanced split and the datasets tested in Section 3 are commonplace benchmarking practices for current SSNC papers [7, 9].

4.1 Comparing Split Performances

Provided in Figures 5-6 are visualizations on the impacts of different evaluation techniques on simple linear models ($\boldsymbol{XW}$ and $\boldsymbol{AXW}$). To keep things comparable to the sparse split, where no validation set exists, both the learning rate (10^{-3}) and the weight decay (0.0) were set to be fixed for the Adam optimizer. Despite this lack of tuning, the best of these models, per dataset, achieve roughly $>85\%$ relative performance when compared to SOTA SSNC methods. The high-end of this performance can be seen in the Squirrel column of Figure 6, where mean accuracy of the best linear model is 77.3%.

New GNN architectures which make use of the recent, balanced split may also experience an analogous performance bump relative to any older models tested before the split was introduced. In the worst case, this may lead to an overstatement in new modeling contributions and has the potential downside of muddying the signal of what makes for a successful and efficient GNN architecture in SSNC experiments. For this reason, we believe it is important to be clear on the impact of splitting conventions and how they contribute to recent performance upticks in SSNC benchmarking.

5 Conclusions

We have shown how classically-inspired, non-parametric techniques can be used to match, and sometimes exceed, previous spectral and non-linear GNN approaches. Our methods make no use of post-model augmentations, such as dropout [14] or batchnorm [15], and allow for a clean theoretical analysis in future work.

Empirically, we explored and ablated pertinent hyperparameters to the spectral kernel model and have shown the various dependences between parameters across different datasets. On the aspect of low-rank kernel models, we have shown how spectral truncation can homogenize response outcomes for different kernel choices. Additionally for low-rank models, we have shown how performance decline is gradual with increases in spectral truncation, pointing to practical speed-ups for non-parametric kernel implementations.

On the aspect of testing conventions, we looked at how evaluation has changed for SSNC tasks since the first introduction of popular citation datasets [20]. We have shown how the class-balanced split can produce improvements in performance outside of what is expected by uncertainty.

In summary, non-parametric kernel aggregators provide a simple yet effective means of recovering unobserved labels in SSNC tasks. As our implementations are free from post-model augmentations, we expect future theoretical insights obtained for low rank kernel aggregators to be closely reproduced in experimental settings such as those seen in Section 3. Future work may further develop these insights, adding to the list of favorable properties for non-parametric kernel aggregators.

References

[1] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh, 2002. 1

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7: 2399–2434, Dec. 2006. 1

[3] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003. 1

[4] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 1

[5] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 1, 2, 8

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021. 5, 9

[7] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and optdoina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 1362–1375, 2022. 1, 5, 9

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc. 1

[9] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23341–23362, 17–23 Jul 2022. 1, 2, 5, 9

[10] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 05 2021. 2, 8

[11] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. 3

[12] Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 3

[13] Stephen J. Young and Edward R. Scheinerman. Random dot product graph models for social networks. In *Algorithms and Models for the Web-Graph*, pages 138–149, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 4

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 4, 9

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. 4, 9

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[17] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. 5, 8, 9

[18] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021. 5

[19] Sudhanshu Chanpuriya and Cameron N Musco. Simplified graph convolution with heterophily. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 5

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008. 7, 9

[21] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 40–48, 2016. 7

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 8

[23] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. AAAI '98/IAAI '98, page 509–516, USA, 1998. 8