arXiv:2301.13764v1 [cs.LG] 31 Jan 2023

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL KERNEL MACHINES

Sonny Achten* **Francesco Tonin** **Panagiotis Patrinos** **Johan A. K. Suykens**

KU Leuven, Department of Electrical Engineering (ESAT),
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
{sonny.achten; francesco.tonin; panos.patrinos; johan.suykens}@esat.kuleuven.be

ABSTRACT

We present a deep Graph Convolutional Kernel Machine (GCKM) for semi-supervised node classification in graphs. First, we introduce an unsupervised kernel machine propagating the node features in a one-hop neighbourhood. Then, we specify a semi-supervised classification kernel machine through the lens of the Fenchel-Young inequality. The deep graph convolutional kernel machine is obtained by stacking multiple shallow kernel machines. After showing that unsupervised and semi-supervised layer corresponds to an eigenvalue problem and a linear system on the aggregated node features, respectively, we derive an efficient end-to-end training algorithm in the dual variables. Numerical experiments demonstrate that our approach is competitive with state-of-the-art graph neural networks for homophilious and heterophilious benchmark datasets. Notably, GCKM achieves superior performance when very few labels are available.

Keywords Graph Neural Networks · Restricted Kernel Machines · GCN · Kernel Methods · Least-Squares Support Vector Machines · Deep Kernel Learning · Node Classification · Semi-Supervised Learning

1 Introduction

Semi-supervised node classification has been an important research area for several years. In many real-life applications, one has structured data for which the entire graph can be observed (e.g., a social network where users are represented as nodes and the relationships between users as edges). However, the node labels can only be observed for a small subset of nodes. The learning task is then to predict the label of unsupervised nodes, given the node attributes of all nodes and the network structure of the graph. In many cases, exploiting the information in a local neighbourhood can boost performance (e.g., friends in the social network are likely to share the same preferences). The challenge has been to exploit this information effectively.

In recent years, graph neural networks (GNNs) have rapidly transformed the field of learning on graphs. Their performance follows from: 1) their ability to effectively propagate the node information through the network iteratively, allowing the learned node representations to capture information from both the node attributes and the network structure; and 2) the end-to-end training, allowing the learning of node representations based on the objective of the final learning task Hamilton [2020], Bacciu et al. [2020], Wu et al. [2021].

More traditionally, kernel-based methods such as support vector machines were the standard in graph learning tasks because of the possibility to use a kernel function that represents pairwise similarities between two graphs as the dot product of their embeddings, without the need to explicitly know these potentially high-dimensional embeddings (i.e., the "kernel-trick") Ghosh et al. [2018], Kriege et al. [2020]. An additional advantage of kernel machines is that they have strong foundations in learning theory and have clear and interpretable optimization Vapnik [1998], Schölkopf and Smola [2002], Suykens et al. [2002]. A drawback of kernel methods, however, is that they do not benefit from hierarchical representation learning as deep learning methods do.

*Corresponding author: sonny.achten@esat.kuleuven.be

On the one hand, some works have explored synergies between graph deep learning methods and graph kernels Lei et al. [2017], Nikolentzos et al. [2018], Du et al. [2019], Feng et al. [2022]. However, these methods are essentially graph neural networks or a shallow kernel machine. Other works have proposed deep versions of kernel machines to combine advantages of both deep learning and kernel methods Cho and Saul [2009], Mairal et al. [2014], Suykens [2017] but no previous work has investigated a deep kernel machine for graph learning.

In this work, we introduce a deep Graph Convolutional Kernel Machine (GCKM) for node classification made of multiple shallow layers with non-parametric aggregation functions. The proposed architecture consists of multiple unsupervised kernel machines for one-hop node aggregation and a final semi-supervised kernel machine. The main idea underlying our method is to combine multiple unsupervised kernel machines such that the final model can achieve higher performance in semi-supervised node classification, where most nodes in the graph are unlabeled. To do so, we show how to train the proposed deep kernel machine in its dual form by directly learning the hidden node representations. Because of the appropriate regularization mechanisms, the neighbourhood aggregation of each layer is implicitly embedded in the final representation, which is a key difference with GNNs. We propose a two-step optimization algorithm with an initialization and fine-tuning phase. Because the model is built on unsupervised core models, augmented with a supervised loss term, we illustrate the possibility to use an unsupervised validation metric. Finally, we show that our model has competitive performance compared to their state-of-the-art GNN counterparts in a transductive node classification setting and outperforms them when very few labels are available, which is of particular interest in many real-world applications where labels are difficult or expensive to collect. In short, we propose the first deep kernel machine for node classification. The reported results can be reproduced using our code in supplementary materials.

2 Preliminaries

This section introduces the notations and definitions which will be used in the remainder of the paper and provides the reader with some important background knowledge.

An undirected and unweighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined by a set of nodes or vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ between these nodes. The node degree is simply the number of adjacent nodes: $d_v = |\mathcal{N}_v|$, where $\mathcal{N}_v$ is the one-hop neighbourhood of node v. As the task of the proposed method will be node classification, we will consider attributed graphs $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ where each node v has a d-dimensional node features vector $\boldsymbol{x}_v$ and a class label y_v. By concatenating the feature vectors, we obtain the node feature matrix $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

We will use lowercase symbols (e.g., x) for scalars, lowercase bold (e.g., $\boldsymbol{x}$) for vectors and uppercase bold (e.g., $\boldsymbol{X}$) for matrices. A single entry of a matrix is represented by X_{ij} where i and j indicate the row and column respectively. Superscripts in brackets indicate the layer in deep architectures whereas subscripts indicate datapoints (e.g., $\boldsymbol{h}_v^{(l)}$). Subscript c indicates a centering, as will be explained. We represent sets with curly brackets $\{\cdot\}$ and use double curly brackets $\{\{\cdot\}\}$ for multisets (i.e., sets that allow multiple instances of a same element). At any point, the reader can consult the list of symbols in appendix A for clarification.

2.1 Graph Neural Networks

GNNs (e.g., Gilmer et al. [2017], Kipf and Welling [2017], Hamilton et al. [2017], Veličković et al. [2018], Xu et al. [2019]) are powerful models for learning with graphs. All GNNs have a permutation equivariant node representation learning part. While for edge and graph level tasks this is followed by an appropriate readout part, no further readout is needed for node level tasks. In a GNN layer, information from a one-hop neighbourhood gets propagated to each node in the graph. By iterating this in successive layers, one increases the receptive field. We refer the interested reader to Bacciu et al. [2020], Wu et al. [2021], Bronstein et al. [2021] for an extensive introduction into the topic.

The majority of GNN layers can be categorized in one of three flavours that differ in the way that they propagate this information through the network Bronstein et al. [2021]: 1) the convolutional flavour, where node features are aggregated, possibly after rescaling them with a given edge weight; 2) the attentional flavour, where this rescaling can be learned through an attention mechanism (e.g., Veličković et al. [2018]); and 3) the message-passing flavour (e.g., Gilmer et al. [2017]), where messages (i.e., vectors) are learned before aggregating them. These flavours are ordered in increasing complexity and generality. We will further focus on the convolutional flavour.

Many convolutional GNN layers can be decomposed into a nonparametric aggregation step $\psi(\cdot, \cdot)$, followed by a nonlinear transformation $\phi(\cdot)$. In this case, the hidden representation of node v in layer l is of the form:

$$\boldsymbol{h}_v^{(l)} = \phi\left(\psi\left(\boldsymbol{h}_v^{(l-1)}, \left\{\left\{\boldsymbol{h}_u^{(l-1)} | u \in \mathcal{N}_v\right\}\right\}\right)\right). \tag{1}$$

Well-known examples are GCN Kipf and Welling [2017]:

$$\boldsymbol{h}_v^{(l)} = \sigma\left(\boldsymbol{W}^T \sum_{u \in \mathcal{N}_v \cup \{v\}} \frac{\boldsymbol{h}_u^{(l-1)}}{\sqrt{\tilde{d}_u \tilde{d}_v}} + \boldsymbol{b}\right),$$

where $\tilde{d}_v$ is the node degree of node v after self-loops were added to the graph, and GIN Xu et al. [2019]:

$$\boldsymbol{h}_v^{(l)} = \text{MLP}^{(l)}\left((1+\epsilon^{(l)}) \cdot \boldsymbol{h}_v^{(l-1)} + \sum_{u \in \mathcal{N}_v} \boldsymbol{h}_u^{(l-1)}\right),$$

which is maximally powerful in the class of message passing neural networks and as expressive as the one-dimensional Weisfeiler-Lehman graph isomorphism test Weisfeiler and Lehman [1968]. Here, $\epsilon^{(l)}$ can be a fixed or a learnable parameter. Xu et al. [2019] have demonstrated that the expressiveness follows from the sum aggregator and the injectiveness of the transformation function, for which they proposed a multilayer perceptron with at least one hidden layer, motivated by the universal approximator theorem Hornik et al. [1989], Hornik [1991].

2.2 Restricted Kernel Machines

Restricted kernel machines (RKMs) Suykens [2017] connect least squares support vector machines (LS-SVMs) and kernel principal component analysis (kernel PCA) with restricted Boltzmann machines Suykens and Vandewalle [1999a], Suykens et al. [2003], Salakhutdinov [2015]. They possess primal and dual model representations based on the concept of conjugate feature duality, which introduces dual variables as hidden features based on an inequality of quadratic forms. The feature map can be defined explicitly (e.g., with a deep neural network) or implicitly by means of a kernel function when using the dual representation.

Analogous to the LS-SVM setting, Suykens [2017] shows that the RKM interpretation of kernel PCA leads to an eigendecomposition of the kernel matrix. Asides kernel PCA, Suykens [2017] also formulated different types of kernel machines in the RKM framework. Deep RKMs are then obtained by stacking multiple RKM layers, where the dual variables are the input for the next layer.

3 Other Related Work

There have been many studies relating kernel machines to deep learning. From a training perspective, Suykens and Vandewalle [1999b] have shown how to train a multilayer perceptron in a LS-SVM setting and more recent works have formulated theoretical connections between training overparametrized (or even infinitely wide) deep neural networks and kernel methods Belkin et al. [2018], Jacot et al. [2018]. With their Convolutional Kernel Networks, Mairal et al. [2014] proposed a multilayer kernel similar to convolutional neural networks, and approximate the kernel function in an unsupervised manner before using it in a shallow kernel machine. Related to graph learning, Du et al. [2019] introduced the graph neural tangent kernel, based on a GNN architecture to train in a shallow kernel machine, whereas Lei et al. [2017] derived a deep learning model based on graph kernels such as the Weisfeiler-Lehman kernel. Both these methods were designed for graph level tasks. On the other hand, Nikolentzos et al. [2018] and Feng et al. [2022] focused on extending GNNs with graph kernels. Although these methods use kernels, they use them as convolutional filters and are in essence not kernel machines.

Generally, kernel machines are closely related to graph learning in the sense that the kernel matrix can be viewed as a similarity graph and vice versa. Belkin et al. [2006] proposed a framework in which both transductive node classification in graphs and semi-supervised classification with support vector machines (for non-structured data) can be derived. Their framework is based on a supervised learning model augmented with a manifold regularization technique, whereas Mehrkanoon et al. [2015] started from an unsupervised objective inspired by spectral clustering of the kernel matrix, and augmented it with a supervised term for the labeled datapoints to tackle the same problem.

In deep kernel learning, the recently proposed RKM framework Suykens [2017] gives a representation in terms of visible and hidden units similar to the Restricted Boltzmann Machine Salakhutdinov [2015]. While RKMs have been successfully applied to unsupervised problems, including generative modelling Pandey et al. [2022], disentangled feature learning Tonin et al. [2021], and multi-view clustering Tao et al. [2022], deep kernel learning for graph learning tasks has not yet been investigated.

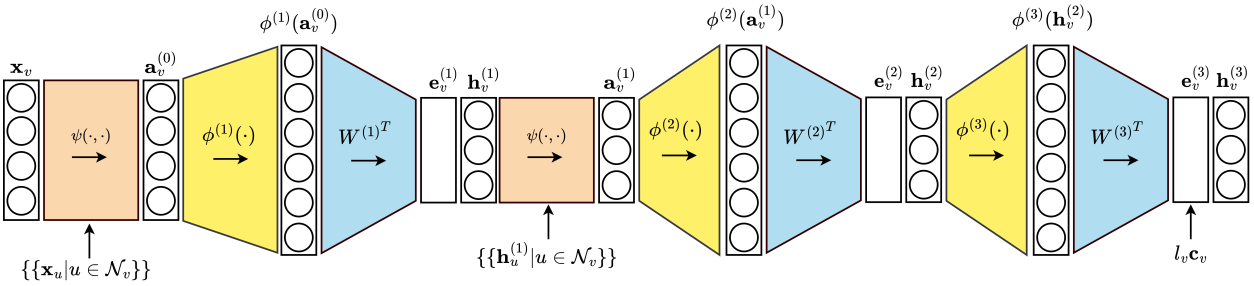


Figure 1: A deep GCKM architecture for semi-supervised node classification, consisting of two GCKM layers (GCKMℓ_1, GCKMℓ_2) and a Semi-SupRKM layer. In each GCKMℓ, the node features are aggregated and then (implicitly) transformed to obtain the error variables. The dual variables are coupled with these error variables by means of conjugate feature duality and serve as input for the next layer. In the final Semi-SupRKM layer, the dual variables are directly used to represent the class labels of the unsupervised nodes.

4 Method

This section introduces the deep Graph Convolutional Kernel Machine (GCKM): a semi-supervised kernel machine propagating information through the network for node classification in graphs. First, the GCKM layer (GCKMℓ) for single-hop propagation is proposed. Also, the semi-supervised kernel machine (Semi-SupRKM) is described. Finally, we explain how to combine these shallow kernel machines in a deep model to increase the receptive field of the model to multiple hops and to perform semi-supervised node classification. All proofs of the lemmas and propositions as well as a full derivation for both the GCKMℓ as the Semi-SupRKM can be found in Appendix B and C, respectively.

4.1 The Graph Convolutional and Semi-Supervised Kernel Machine as Building Blocks

Graph Convolutional Kernel Machine GCKMℓ is based on the RKM interpretation of kernel PCA as introduced by Suykens [2017]. However, the node features are aggregated before they are mapped into the feature space. We start from the primal minimization problem:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{e}_v} \quad & J = \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}) - \frac{1}{2}\sum_{v=1}^{n} \boldsymbol{e}_v^T\boldsymbol{\Lambda}^{-1}\boldsymbol{e}_v \\ \text{subject to} \; & \begin{cases} \boldsymbol{e}_v = \boldsymbol{W}^T\phi_c(\boldsymbol{a}_v), \quad v = 1,\ldots,n \\ \boldsymbol{a}_v = \psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) \end{cases}, \end{aligned} \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_f \times s}$ is an unknown interconnection matrix, $\boldsymbol{e}_v \in \mathbb{R}^s$ the error variables, $n = |\mathcal{V}_{\text{tr}}|$ the number of training nodes, and symmetric hyperparameter matrix $\boldsymbol{\Lambda} \succ 0$. Given a feature map $\phi(\cdot)$, the centered feature map is defined as $\phi_c(\cdot) \triangleq \phi(\cdot) - \Sigma_i\phi(\boldsymbol{x}_i)/n$. Note that the formulation of the error variables has the same form as (1) with $\boldsymbol{W}^T\phi_c(\cdot)$ as the transformation step. Because of the minus sign in the objective function, one can interpret this minimization problem conceptually as maximizing the variance of the error variables $\boldsymbol{e}_i$ around zero target, while keeping the weights $\boldsymbol{W}$ small Suykens et al. [2003].

Dual variables $\boldsymbol{h}_i$ are introduced using a simple case of Fenchel-Young inequality Rockafellar [1974]:

$$\frac{1}{2}\boldsymbol{e}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{e} + \frac{1}{2}\boldsymbol{h}^T\boldsymbol{\Lambda}\boldsymbol{h} \geq \boldsymbol{e}^T\boldsymbol{h}, \quad \forall \boldsymbol{e},\boldsymbol{h} \in \mathbb{R}^s, \forall \boldsymbol{\Lambda} \in \mathbb{R}^{s\times s}_{\succ 0}.$$

When substituting the above in (2) and eliminating the error variables, one obtains a primal-dual minimization problem as an upper bound on the primal objective function:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{h}_v} \bar{J} \triangleq & -\sum_{v=1}^{n} \phi_c(\boldsymbol{a}_v)^T\boldsymbol{W}\boldsymbol{h}_v \\ & + \frac{1}{2}\sum_{v=1}^{n} \boldsymbol{h}_v^T\boldsymbol{\Lambda}\boldsymbol{h}_v + \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}). \end{aligned} \tag{3}$$

Note that problem (3) is generally nonconvex. Whether or not it has a solution depends on hyperparameters $\boldsymbol{\Lambda}$. In the next lemma we show how to determine $\boldsymbol{\Lambda}$ automatically by the optimization. We next define the gram matrix $\boldsymbol{K}$ with $K_{uv} = \phi(\boldsymbol{a}_u)^T\phi(\boldsymbol{a}_v)$, which depends on the aggregated node features; $\boldsymbol{K}_c = \boldsymbol{M}_c\boldsymbol{K}\boldsymbol{M}_c$ with $\boldsymbol{M}_c = (\boldsymbol{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T)$ the centering matrix; and $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n]^T$.

Lemma 4.1. *The solution to the dual minimization problem:*

$$\min_{\boldsymbol{H}} -\frac{1}{2\eta} Tr(\boldsymbol{H}^T\boldsymbol{K}_c(\boldsymbol{X}, \mathcal{E})\boldsymbol{H}) \quad \textit{subject to } \boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I}_s, \tag{4}$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ *as* (3) *when the hyperparameters* $\boldsymbol{\Lambda}$ *in* (3) *are chosen to equal the symmetric part of the Lagrange multipliers* $\boldsymbol{Z}$ *of the equality constraints in* (4)*; i.e.,* $\boldsymbol{\Lambda} = (\boldsymbol{Z} + \boldsymbol{Z}^T)/2$.

Now, (4) is bounded and is guaranteed to have a minimizer. Indeed, it is a minimization of a concave objective over a compact set. Note that (4) can be solved by a gradient-based algorithm. The solution satisfies the following property:

Remark 4.2. Given a symmetric matrix $\boldsymbol{K}_c$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_s > \lambda_{s+1} \geq \cdots \geq \lambda_n \geq 0$, and $\eta > 0$ a hyperparameter; and let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s$ be the columns of $\boldsymbol{H}$; then $\boldsymbol{H}$ is a minimizer of (4) if and only if $\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I}_s \wedge \text{span}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s) = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$, where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ are the eigenvectors of $\boldsymbol{K}_c$ corresponding to the s largest eigenvalues.

Remark 4.3. One can obtain a solution of (4) by solving the eigendecomposition problem:

$$\frac{1}{\eta}\boldsymbol{K}_c(\boldsymbol{X}, \mathcal{E})\boldsymbol{H} = \boldsymbol{H}\boldsymbol{\Lambda}, \tag{5}$$

and selecting the eigenvectors corresponding to the s largest eigenvalues as the columns of $\boldsymbol{H}$.

Notice that (5) is the kernel PCA formulation, a nonlinear generalization of PCA Schölkopf and Smola [2002], Suykens et al. [2003], with the aggregated node features as the input, and where the first s components represent the data. The solution of (4) generally yields any orthonormal basis for the same subspace as spanned by the first s components, and therefore embeds the same information in the dual representations. Further, instead of explicitly defining a feature map, one can apply the kernel trick using Mercer's theorem, stating that for any positive definite kernel $k(\cdot, \cdot)$ there exist a, possibly infinite dimensional, feature map $\phi(\cdot)$ such that $\phi(\boldsymbol{a}_u)^T\phi(\boldsymbol{a}_v) = k(\boldsymbol{a}_u, \boldsymbol{a}_v)$ Mercer [1909]. In this case, the transformation function $\boldsymbol{W}^T\phi(\cdot)$ is only implicitly defined. As the kernel function, one could choose for example a linear kernel[2], a polynomial kernel[3], or a radial basis function (RBF)[4], among others. We can now define the GCKM layer:

Definition 4.4 (Graph Convolutional Kernel Machine layer)**.** GCKMℓ is a kernel machine for unsupervised node representation learning that propagates information through the network in a one-hop neighbourhood in a convolutional flavour. More formally, it can be interpreted as a principal component analysis on the aggregated node features in a kernel induced feature space, where the latent representations are obtained by solving either (4) or (5), and are used as the input for the subsequent layer in a deep GCKM.

For the aggregation step, we can choose any function that can handle multisets of different sizes and that is invariant to permutations on this multiset. In our experiments, we use GCN aggregation:

$$\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) = \sum_{u \in \mathcal{N}_v \cup \{v\}} \frac{\boldsymbol{x}_u}{\sqrt{\tilde{d}_u \tilde{d}_v}}, \tag{6}$$

as well as sum aggregation:

$$\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) = \boldsymbol{x}_v + \sum_{u \in \mathcal{N}_v} \boldsymbol{x}_u. \tag{7}$$

We proceed with some properties of GCKMℓ. The model-based approach of kernel PCA gives us the possibility to use one set of nodes for training $\mathcal{V}_{\text{tr}}$ and use an out-of-sample extension for another (super)set of nodes $\mathcal{V}$, possibly from another graph. The following result follows from the stationarity conditions:

[2] $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$
[3] $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T\boldsymbol{x}_j + t)^p$
[4] $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/(2\sigma^2)\right)$

Lemma 4.5. *Let $n = |\mathcal{V}_{tr}|$, $m = |\mathcal{V}|$, and $\boldsymbol{K}^{\mathcal{V}_1,\mathcal{V}_2} \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$ a kernel matrix containing kernel evaluations of all elements of set $\mathcal{V}_1$ w.r.t. all elements of set $\mathcal{V}_2$ (i.e., $K^{\mathcal{V}_1,\mathcal{V}_2}_{uv} = k(\boldsymbol{a}_u, \boldsymbol{a}_v)\ \forall u \in \mathcal{V}_1, \forall v \in \mathcal{V}_2$). The error variables can then be obtained using:*

$$\hat{\boldsymbol{E}}_{\mathcal{V}} = \frac{1}{\eta}\boldsymbol{K}^{\mathcal{V},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}} - \frac{\mathbf{1}_m\mathbf{1}_n^T\boldsymbol{K}^{\mathcal{V}_{tr},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}}}{n\eta}, \tag{8}$$

where $\hat{\boldsymbol{E}}_{\mathcal{V}} = [\hat{e}_1, \ldots, \hat{e}_m]^T$. The hidden representations are given by $\hat{\boldsymbol{H}}_{\mathcal{V}} = \hat{\boldsymbol{E}}_{\mathcal{V}}\boldsymbol{\Lambda}^{-1}$.

Equation (8) is useful for large-scale problems, when subsets are used for training, and for inductive tasks. It also satisfies the permutation equivariance condition.

Proposition 4.6. *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, the aggregated node features $\{a_v : v \in \mathcal{V}_{tr}\}$ and latent representations $\boldsymbol{H}_{\mathcal{V}_{tr}}$of the training nodes $\mathcal{V}_{tr}$, and a local aggregation function $\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\})$ that is permutation invariant; the mapping f from $\mathcal{G}$ to $\mathcal{G}' = (\mathcal{V}, \mathcal{E}, \hat{\boldsymbol{E}}_{\mathcal{V}})$ using (8) is equivariant w.r.t. any permutation $\pi(\mathcal{G})$, i.e., $\mathcal{G}' = f(\mathcal{G}) \iff \pi(\mathcal{G}') = f(\pi(\mathcal{G}))$.*

A theoretical analysis of the expressiveness of GCKMℓ can be put in context of the theoretical results by Xu et al. [2019]. Any associated feature map of the RBF-kernel is injective Christmann and Steinwart [2008]. Furthermore, it has been established that SVMs using the RBF-kernel are universal approximators Burges [1998], Hammer and Gersmann [2003].

Lemma 4.7. *A GCKMℓ that uses sum aggregation and a RBF-kernel is as expressive as an iteration of the Weisfeiler-Lehman graph isomorphism test Weisfeiler and Lehman [1968].*

Semi-Supervised Restricted Kernel Machine Next, we introduce a multi-class semi-supervised kernel machine for classification (Semi-SupRKM). Like Mehrkanoon et al. [2015], we start from kernel spectral clustering as an unsupervised core model, and augment it with a supervised loss term. Here however, to be able to use it in a deep kernel machine, we introduce duality with conjugated features, rather than by means of Lagrange multipliers.

The primal minimization problem is given by:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{e}_i,\boldsymbol{b}} \quad J = \frac{\eta}{2}\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{W}) - \frac{1}{2\lambda_1}\sum_{i=1}^{n} v_i\boldsymbol{e}_i^T\boldsymbol{e}_i \\ + \frac{1}{2\lambda_2}\sum_{i=1}^{n} l_i(\boldsymbol{e}_i - \boldsymbol{c}_i)^T(\boldsymbol{e}_i - \boldsymbol{c}_i)\end{aligned} \tag{9}$$

$$\text{subject to} \quad \boldsymbol{e}_i = \boldsymbol{W}^T\phi(\boldsymbol{x}_i) + \boldsymbol{b}, \quad i = 1, \ldots, n, \tag{10}$$

where $l_i \in \{0, 1\}$ indicates whether the label of datapoint i is used in training, $\boldsymbol{c}_i \in \{-1, 1\}^p$ codes its class label (e.g., in a one-vs-all encoding[5]), and v_i is a weighting scalar obtained as the inverse degree of the datapoint in the similarity graph defined by $\boldsymbol{K} = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e., $v_i = 1/\Sigma_j K_{ij}$.

By introducing Fenchel-Young inequalities:

$$-\frac{1}{2}\boldsymbol{e}_i^T\boldsymbol{e}_i \leq \frac{1}{2}\boldsymbol{h}_i^T\boldsymbol{h}_i - \boldsymbol{e}_i^T\boldsymbol{h}_i$$

$$-\frac{1}{2}(\boldsymbol{e}_i - \boldsymbol{c}_i)^T(\boldsymbol{e}_i - \boldsymbol{c}_i) \leq \frac{1}{2}\boldsymbol{h}_i^T\boldsymbol{h}_i - (\boldsymbol{e}_i - \boldsymbol{c}_i)^T\boldsymbol{h}_i,$$

and defining $r_i = \frac{v_i}{\lambda_1} - \frac{l_i}{\lambda_2}$, one obtains the primal-dual minimization problem as an upper bound on the primal objective function:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{h}_i,\boldsymbol{b}} \bar{J} \triangleq \frac{\eta}{2}\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{W}) + \frac{1}{2}\sum_{i=1}^{n} r_i\boldsymbol{h}_i^T\boldsymbol{h}_i \\ - \sum_{i=1}^{n} r_i(\boldsymbol{W}^T\phi(\boldsymbol{x}_i) + \boldsymbol{b})^T\boldsymbol{h}_i - \sum_{i=1}^{n}\frac{l_i}{\lambda_2}\boldsymbol{c}_i^T\boldsymbol{h}_i.\end{aligned} \tag{11}$$

We next define matrices $\boldsymbol{R} = \mathrm{diag}(r_1, \ldots, r_n)$; $\boldsymbol{L} = \mathrm{diag}(l_1, \ldots, l_n)$; $\boldsymbol{S} = \boldsymbol{I}_n - \frac{\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{R}}{\mathbf{1}_n^T\boldsymbol{R}\mathbf{1}_n}$; $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n]^T$; and $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n]^T$.

[5] A one-vs-all encoding is similar to a one-hot encoding but it uses -1 instead of 0.

Lemma 4.8. *The solution to the dual minimization problem:*

$$\begin{aligned}\min_{\boldsymbol{H}} & -\frac{1}{2\eta} Tr(\boldsymbol{H}^T \boldsymbol{R}\boldsymbol{K}(\boldsymbol{X})\boldsymbol{R}\boldsymbol{H}) \\ & + \frac{1}{2} Tr(\boldsymbol{H}^T \boldsymbol{R}\boldsymbol{H}) - \frac{1}{\lambda_2} Tr(\boldsymbol{H}^T \boldsymbol{L}\boldsymbol{C}) \\ \textit{subject to } & \boldsymbol{H}^T \boldsymbol{R}\mathbf{1}_n = \mathbf{0}_p \end{aligned} \tag{12}$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ *as* (11) *where the Lagrange multipliers equal the bias* $\boldsymbol{b}$*.*

Remark 4.9. Alternatively, one can find the dual variables by solving a linear system in the dual variables:

$$(\boldsymbol{I}_n - \frac{1}{\eta}\boldsymbol{R}\boldsymbol{S}\boldsymbol{K}(\boldsymbol{X}))\boldsymbol{R}\boldsymbol{H} = \frac{1}{\lambda_2}\boldsymbol{S}^T\boldsymbol{L}\boldsymbol{C}. \tag{13}$$

From the stationarity conditions, one obtains $\boldsymbol{e}_i = \boldsymbol{h}_i - \frac{l_i \boldsymbol{c}_i}{r_i \lambda_2}$, which simplifies to $\boldsymbol{e}_i = \boldsymbol{h}_i$ for the unsupervised training points. One can thus directly infer the class label $\hat{y}_i$ from the learned representation by comparing the class codes and select the one with closest Hamming distance to the error variable $\boldsymbol{e}_i$. For one-vs-all encoding, this is simply the index with highest value: $\hat{y}_i = \text{argmax}_j (\boldsymbol{h}_i)_j$. When using a subsample for training, one can use the out-of-sample extension (26) described in Appendix C.

4.2 Deep Graph Convolutional Kernel Machine

Next, we construct a deep graph convolutional kernel machine for semi-supervised node classification by stacking multiple GCKMℓ's and a Semi-SupRKM (Figure 1). Similar to GNNs, the dual variables of the GCKMℓ's ($\boldsymbol{H}^{(1)}$ and $\boldsymbol{H}^{(2)}$) serve as input for the subsequent layer and can thus be viewed as hidden representations. The dual variables of the Semi-SupRKM layer ($\boldsymbol{H}^{(3)}$), can directly be used to infer the class label of the unlabeled nodes. The optimization problem for end-to-end learning is given by combining the dual minimization problems of the different layers (i.e., (4) and (12)). For two GCKM layers and a Semi-SupRKM layer, this yields:

$$\begin{aligned}\min_{\boldsymbol{H}^{(1)},\boldsymbol{H}^{(2)},\boldsymbol{H}^{(3)}} J_{\text{GCKM}} \triangleq & -\frac{1}{2\eta^{(1)}}\text{Tr}(\boldsymbol{H}^{(1)^T}\boldsymbol{K}_c^{(1)}\boldsymbol{H}^{(1)}) \\ & -\frac{1}{2\eta^{(2)}}\text{Tr}(\boldsymbol{H}^{(2)^T}\boldsymbol{K}_c^{(2)}\boldsymbol{H}^{(2)}) \\ & -\frac{1}{2\eta^{(3)}}\text{Tr}(\boldsymbol{H}^{(3)^T}\boldsymbol{R}\boldsymbol{K}^{(3)}\boldsymbol{R}\boldsymbol{H}^{(3)}) \\ & +\frac{1}{2}\text{Tr}(\boldsymbol{H}^{(3)^T}\boldsymbol{R}\boldsymbol{H}^{(3)}) \\ & +\frac{1}{\lambda_2^{(3)}}\text{Tr}(\boldsymbol{H}^{(3)^T}\boldsymbol{L}\boldsymbol{C}) \\ \text{subject to} & \begin{cases} \boldsymbol{H}^{(1)^T}\boldsymbol{H}^{(1)} = \boldsymbol{I}_{s_1} \\ \boldsymbol{H}^{(2)^T}\boldsymbol{H}^{(2)} = \boldsymbol{I}_{s_2} \\ \boldsymbol{H}^{(3)^T}\boldsymbol{R}\mathbf{1}_n = \mathbf{0}_p. \end{cases} \end{aligned} \tag{14}$$

Like in GNNs, the number of GCKMℓ's used in the deep GCKM determines the receptive field of the model (i.e., the number of hops that the information propagates through the network). However, the key difference is that in GCKM, this message passing is implicitly embedded in the final representation.

4.3 Training Deep Graph Convolutional Kernel Machines

In the algorithmic aspect, the constrained optimization problem (14) is addressed with an alternating minimization scheme, as shown in Algorithm 1. First, note that the constraint set for the two GCKM layers is the Stiefel manifold $\text{St}(s_j, n) = \{\boldsymbol{H}^{(j)} \in \mathbb{R}^{n \times s_j} \mid \boldsymbol{H}^{(j)^\top}\boldsymbol{H}^{(j)} = \boldsymbol{I}_{s_j}\},\ j = 1, 2$. We therefore employ the Cayley Adam optimizer Li et al. [2019] to update $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ with $\boldsymbol{H}^{(3)}$ fixed. Then, $\boldsymbol{H}^{(3)}$ is updated by solving the linear system (13) associated with the semi-supervised layer.

Similar to stacked auto-encoders Bengio [2009], the dual variables $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ and $\boldsymbol{H}^{(3)}$ are initialized by sequentially solving the layers as individual kernel machines before finetuning end-to-end. $\boldsymbol{H}_0^{(1)}$ is thus obtained by solving (5),

Algorithm 1 Optimization algorithm of GCKM.

Input: Graph with node features $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, label encodings of training points $\boldsymbol{LC}$
Output: $\boldsymbol{H}_T^{(1)}, \boldsymbol{H}_T^{(2)}, \boldsymbol{H}_T^{(3)}$
Initialize $\{\boldsymbol{H}_0^{(1)}, \boldsymbol{H}_0^{(2)}, \boldsymbol{H}_0^{(3)}\}$
for $k \leftarrow 0, 1, \ldots, T$ **do**
    Compute $\boldsymbol{K}_c^{(1)}$ from aggregated $\boldsymbol{X}$
    Compute $\boldsymbol{K}_c^{(2)}$ from aggregated $\boldsymbol{H}_k^{(1)}$
    Update $\{\boldsymbol{H}_{k+1}^{(1)}, \boldsymbol{H}_{k+1}^{(2)}\} \leftarrow$ CayleyAdam(J_{GCKM})
    Compute $\boldsymbol{K}^{(3)}$ from $\boldsymbol{H}_{k+1}^{(2)}$
    Update $\{\boldsymbol{H}_{k+1}^{(3)}\} \leftarrow$ Solve (13) with $\boldsymbol{K}^{(3)}$
end for

with the initial node features as the input. Then $\boldsymbol{H}_0^{(2)}$ is obtained by solving (5) with $\boldsymbol{H}_0^{(1)}$ as the input node features. Finally, $\boldsymbol{H}_0^{(3)}$ is obtained by solving (13) with $\boldsymbol{H}_0^{(2)}$ as the input node features.

As a validation metric, one can use the accuracy of the validation set or a different supervised metric. Alternatively, because the core model of the Semi-SupRKM is based on kernel spectral clustering, one can use an unsupervised metric that quantifies the quality of obtained clustering Langone et al. [2013]. For node v, the centered cosine distance w.r.t. class s is:

$$d_{v,s}^{\cos} = 1 - \frac{(\boldsymbol{c}_s - \boldsymbol{\mu})^T(\boldsymbol{e}_v - \boldsymbol{\mu})}{||\boldsymbol{c}_s - \boldsymbol{\mu}||\,||\boldsymbol{e}_v - \boldsymbol{\mu}||}, \tag{15}$$

where $\boldsymbol{c}_s$ is the coding of class s and $\boldsymbol{\mu}$ is the center of all codings.[6] The unsupervised performance metric for nodes $\mathcal{V}_{\text{unsup}}$ is then obtained by assigning each node to its closest class encoding and averaging the cosine distances:

$$\mathcal{L}_{\text{unsup}} = \frac{1}{|\mathcal{V}_{\text{unsup}}|} \sum_{v=1}^{|\mathcal{V}_{\text{unsup}}|} \min_s d_{v,s}^{\cos}. \tag{16}$$

5 Experiments

In this section, we assess our model by conducting experiments in a semi-supervised node classification setting on a variety of open graph benchmark datasets and compare its performance with that of several graph neural network baselines. We also assess the performance of the model in different settings in which the unsupervised validation metric is used.

5.1 Datasets and main setting

As datasets, we use three homophilious graphs: Cora, CiteSeer, and PubMed Sen et al. [2008], Yang et al. [2016], which are citation graphs, as well as five heterophilious graphs: Chameleon and Squirrel (Wikipedia graphs Rozemberczki et al. [2021]), webpage graphs Cornell and Texas (WebKB[7]; Pei et al. [2020]), and the Actor co-occurrence graph Tang et al. [2009], Pei et al. [2020]. Table 1 summarizes the dataset statistics, in which $\mathcal{H}(\mathcal{G}) = \frac{1}{n}\sum_{v\in\mathcal{V}} \frac{|\{u\in\mathcal{N}_v : y_u = y_v\}|}{|\mathcal{N}_v|}$ is a measure for the level of homophily of nodes in the graph Pei et al. [2020].

All graphs are undirected and unweighted and we use the same experimental setup as He et al. [2022], including the random seeds for the datasplits. We used the standard fixed splits. For Cora, CiteSeer, and PubMed, there are 20 training labels per class, 500 labels for validation, and 1000 labels for testing. For the other datasets a 2.5%/2.5%/95% train/validation/test-split is used.

We trained a deep GCKM with two GCKM layers and one Semi-SupRKM layer, as depicted in Figure 1, which we will simply refer to as GCKM. We used GCN aggregation (6) for the citation networks and sum aggregation (7) for the heterophilious graphs. We compare our method to a multilayer perceptron (MLP) and to GCN Kipf and Welling [2017] as it is the most comparable GNN counterpart to our method. Furthermore, we add APPNP Klicpera et al. [2019],

[6] For one-vs-all encoding, this becomes $\boldsymbol{\mu} = \mathbf{1}_p/p$.

[7] cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb

Table 1: Dataset statistics.

DATASET	NODES	EDGES	FEATURES	CLASSES	$\mathcal{H}(\mathcal{G})$
CHAM.	2,277	31,371	2,325	5	0.25
SQUI.	5,201	198,353	2,089	5	0.22
TEXAS	183	279	1,703	5	0.06
CORN.	183	277	1,703	5	0.30
ACTOR	7,600	26,659	932	5	0.22
CORA	2,708	5,278	1,433	7	0.83
CITE.	3,327	4,552	3,703	6	0.72
PUBM.	19,717	44,324	500	3	0.79

Table 2: Mean test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification with fixed splits where fewer labels are given. The best model is highlighted in bold and the next best model is underlined for each dataset. Since GCKM has a deterministic training procedure, no confidence intervals are reported.

METHOD	CHAM.	SQUI.	ACTOR	CORA	CITE.	PUBM.
MLP	22.20±1.22	23.50±2.33	23.80±1.62	40.95±1.47	50.05±1.38	68.65±0.59
GCN	25.09±3.05	23.22±2.37	23.67±0.27	76.70±3.41	64.29±0.96	76.68±0.36
APPNP	25.07±2.50	22.59±2.29	23.45±0.30	80.06±0.45	66.46±0.92	77.18±0.46
GPR-GNN	29.58±3.06	24.93±3.57	24.43±0.57	80.16±0.66	64.58±1.27	76.95±2.96
BERNNET	29.17±3.07	24.59±2.59	**24.93±1.68**	80.68±0.62	64.88±1.03	**77.51±0.33**
CHEBNETII	30.07±0.83	24.58±2.50	23.68±0.58	78.86±0.55	67.26±0.68	74.84±0.76
GCKM	**30.17**	**25.38**	23.83	**80.74**	**67.53**	75.10

BernNet He et al. [2021], GPR-GNN Chien et al. [2021], and ChebNetII He et al. [2022] to the comparison because all these methods achieve state-of-the-art performance on at least one of the datasets. The reader can consult Appendix D for more details about the hyperparameter search.

5.2 Semi-Supervised Node Classification with Fixed Splits and Few Labels

In the first experiment, we assess the models in case fewer labels are available. We decrease the total number of labels that are available for training and validation in the standard fixed splits by a factor five. This means that for Cora, CiteSeer and PubMed a total of 4 labels per class with an additional 100 labels are used. For Chameleon, Squirrel, and Actor, 1% of the node labels are used. Texas and Cornell are not part of this experiment, as these datasets are too small. For GCKM, all labels are used for training and the model is selected based on the cosine similarity metric (16) after a random search. For the baseline models, we use a 5-fold crossvalidation[8] scheme and take the best model based on the average validation accuracy over the 5 folds from a grid search. The results are reported in table 2. We observe that GCKM achieves overall superior performance, with highest accuracies on 4 out of 6 datasets: Chameleon, Squirrel, Cora, and CiteSeer.

5.3 Semi-Supervised Node Classification with Standard Fixed Splits

The next experiment uses the standard fixed splits. For each dataset, we performed a random search to determine the hyperparameters and selected the model with highest validation accuracy. For the baseline models, we use the mean test accuracies and 95% confidence intervals as reported in He et al. [2022]. Table 3 summarizes the results.

We observe that for Cornell and Actor, a simple MLP outperforms all models except BernNet and ChebNetII. Comparing to GCN, we observe that for Squirrel and CiteSeer the performance of GCKM is similar, whereas for all other datasets, GCKM again outperforms its GNN counterpart. Comparing our model to the models with more advanced aggregation techniques, we see that GCKM achieves second best performance on Chameleon and Squirrel and reaches new state-of-the-art performance on Texas and Cora.

[8]We use 5-fold crossvalidation because the validation sets might be too small in a 10-fold.

Table 3: Mean test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification with fixed splits.

METHOD	CHAM.	SQUI.	TEXAS	CORN.	ACTOR	CORA	CITE.	PUBM.
MLP	21.91±2.11	23.42±0.94	45.03±2.45	46.18±5.10	29.16±0.52	58.88±0.62	56.97±0.54	73.15±0.28
GCN	39.14±0.60	30.06±0.75	32.42±2.23	35.57±3.55	21.96±0.54	81.32±0.18	71.77±0.21	79.15±0.18
APPNP	30.06±0.96	25.18±0.35	46.31±3.01	45.73±4.85	28.19±0.31	83.52±0.24	72.09±0.25	80.23±0.15
GPR-GNN	30.56±0.94	25.11±0.51	45.76±3.78	43.42±4.95	27.32±0.83	83.95±0.22	70.92±0.57	78.97±0.27
BERNNET	26.35±1.04	24.57±0.72	48.21±3.17	46.64±5.62	29.27±0.23	83.15±0.32	72.24±0.25	79.65±0.25
CHEBNETII	**46.45±0.53**	**36.18±0.46**	54.68±3.87	**50.92±5.49**	**29.54±0.46**	83.67±0.33	**72.75±0.16**	**80.48±0.23**
GCKM	41.16	30.10	**56.07**	38.73	26.01	**84.20**	71.80	80.10

Table 4: Test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification on Cora, CiteSeer and PubMed with fixed split for smaller validation set sizes. The best performing model is highlighted in bold.

	METHOD	VALIDATION LABELS PER CLASS 0	1	5	10
CORA	GCN	-	81.18±0.66	79.50±0.56	80.51±0.25
	CHEBNETII	-	60.15±2.13	77.67±1.03	79.79±1.07
	GCKM	**82.40**	**82.40**	**82.40**	**81.90**
CITE.	GCN	-	63.91±0.95	67.71±1.30	**69.22±1.07**
	CHEBNETII	-	51.52±4.40	67.72±1.06	68.13±1.14
	GCKM	**68.10**	**68.10**	**68.10**	68.10
PUBM.	GCN	-	75.43±0.41	76.42±0.79	74.52±0.53
	CHEBNETII	-	63.31±2.30	**76.79±0.95**	71.65±0.70
	GCKM	**76.80**	**76.60**	76.60	**76.80**

5.4 Semi-Supervised Node Classification with Fixed Splits and Few Validation Labels

This experiment repeats the above experiment for Cora, CiteSeer and PubMed, again with the same 20 training labels per class and 1000 test labels, but the experiment is repeated for different validation sets that are increasing in size. We performed a random search for the hyperparameters and reported in each run the the unsupervised metric $\mathcal{L}_{\text{unsup}}$ (16) on the test set, as well as the validation accuracies $\mathcal{L}_{\text{val}}$ for each validation set. For each validation set, we selected the model that had highest combined score: $\mathcal{L}_{\text{comb}} = (|\mathcal{V}_{\text{val}}|\mathcal{L}_{\text{val}} + |\mathcal{V}_{\text{test}}|\mathcal{L}_{\text{unsup}})/(|\mathcal{V}_{\text{val}}| + |\mathcal{V}_{\text{test}}|)$. We trained the baseline models using the code provided by He et al. [2021] and He et al. [2022], did a hyperparameter gridsearch and selected the model with highest validation accuracy. Table 4 shows the resulting performances of the experiment.

When fewer validation labels are given than in the previous experiment, we see that GCKM achieves best performance on all but two cases. Even with no validation labels, when only the unsupervised metric is used, GCKM is able to obtain a decent performing model, whereas it is not possible to do early stopping or even select a model with the other methods. We conclude that GCKM is less sensitive to a decrease in validation set size than the baseline methods.

5.5 Ablation Studies

We refer the interested reader to Appendix E for an empirical analysis of the effects of the message passing, the training progress, and the significance of layerwise initialization. Also the computational complexity of the training procedure is discussed.

6 Conclusion

We introduce GCKM, a new approach for message passing in graphs based on a deep kernel machine with convolutional aggregation functions. Through the use of duality, we derive optimization problems for the unsupervised and semi-supervised building blocks and we show optimization algorithms for end-to-end training for the semi-supervised node classification task. Numerical experiments on several benchmark datasets verify the effectiveness of the proposed method. Thanks to the unsupervised core, our model outperforms current state-of-the-art GNNs for semi-supervised

node classification when few labels are available for training, which can be a considerable advantage in real-world applications. Many directions for future work exist, such as extending the method to inductive tasks or attentional message passing and investigating methods for scaling up the kernel machines to very large graphs.

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. This work was supported in part by the KU Leuven Research Council (Optimization frameworks for deep kernel machines C14/18/068); the Flemish Government FWO projects GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant; Flemish Government (AI Research Program). Johan Suykens, Sonny Achten, Francesco Tonin and Panagiotis Patrinos are also affiliated with Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium.

References

William L. Hamilton. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A Gentle Introduction to Deep Learning for Graphs. *Neural Networks*, 129:203–221, 2020.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

Swarnendu Ghosh, Nibaran Das, Teresa Gonçalves, Paulo Quaresma, and Mahantapas Kundu. The journey of graph kernels through two decades. *Computer Science Review*, 27:88–111, 2018.

Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5 (1):1–42, 2020.

V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving Neural Architectures from Sequence and Graph Kernels. In *ICML*, 2017.

Giannis Nikolentzos, Polykarpos Meladianos, Antoine Jean-Pierre Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. Kernel Graph Convolutional Neural Networks. In *ICANN*, 2018.

Simon S. Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *NIPS*, 2019.

Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. KerGNNs: Interpretable Graph Neural Networks with Graph Kernels. In *AAAI*, 2022.

Youngmin Cho and Lawrence K Saul. Kernel Methods for Deep Learning. In *NIPS*, 2009.

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional Kernel Networks. In *NIPS*, 2014.

Johan A.K. Suykens. Deep Restricted Kernel Machines Using Conjugate Feature Duality. *Neural Computation*, 29(8): 2123–2163, 2017.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, 2017.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *CoRR*, abs/2104.13478, 2021. URL `https://arxiv.org/abs/2104.13478`.

Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Johan A.K. Suykens and Joos Vandewalle. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999a.

Johan A.K. Suykens, Tony Van Gestel, Joos Vandewalle, and Bart De Moor. A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, 14(2):447–450, 2003.

Ruslan Salakhutdinov. Learning Deep Generative Models. *Annual Review of Statistics and Its Application*, 2(1): 361–385, 2015.

Johan A.K. Suykens and Joos Vandewalle. Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, 10(4):907–911, 1999b.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *ICML*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *NIPS*, 2018.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

S. Mehrkanoon, C. Alzate, R. Mall, R. Langone, and Johan A. K. Suykens. Multiclass Semisupervised Learning Based Upon Kernel Spectral Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(4):20–33, 2015.

Arun Pandey, Michaël Fanuel, Joachim Schreurs, and Johan A. K. Suykens. Disentangled Representation Learning and Generation With Manifold Optimization. *Neural Computation*, 34(10):2009–2036, 2022.

Francesco Tonin, Panagiotis Patrinos, and Johan A. K. Suykens. Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints. *Neural Networks*, 142:661–679, 2021.

Qinghua Tao, Francesco Tonin, Panagiotis Patrinos, and Johan A. K. Suykens. Tensor-based multi-view spectral clustering via shared latent space, 2022. URL `https://arxiv.org/abs/2207.11559`.

R. T. Rockafellar. *Conjugate Duality and Optimization*. SIAM, 1974.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, Series A*, 209(441-458):415–446, 1909.

Andreas Christmann and Ingo Steinwart. *Support Vector Machines*. Springer New York, NY, 2008.

Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

Barbara Hammer and Kai Gersmann. A Note on the Universal Approximation Capability of Support Vector Machines. *Neural Processing Letters*, 17:43–53, 2003.

Jun Li, Fuxin Li, and Sinisa Todorovic. Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform. In *ICLR*, 2019.

Y. Bengio. Learning Deep Architectures for AI. *Foundations*, 2:1–55, 01 2009. doi:10.1561/2200000006.

R. Langone, R. Mall, and J. A. K. Suykens. Soft Kernel Spectral Clustering. In *IJCNN*, 2013.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI magazine*, 29(3):93–93, 2008.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*, 2016.

Benedek Rozemberczki, Carl Allen, Rik Sarkar, and xx Thilo Gross. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(1):1–22, 2021.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*, 2020.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social Influence Analysis in Large-scale Networks. In *KDD*, 2009.

Mingguo He, Zhewei Wei, and Ji-Rong Wen. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. In *NIPS*, 2022.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*, 2019.

Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. In *NIPS*, 2021.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*, 2021.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

A List of symbols

Symbol	Space	Description
$\mathbf{0}_n$	$\{0\}^n$	n-dimensional vector of all zeros
$\mathbf{1}_n$	$\{1\}^n$	n-dimensional vector of all ones
δ_{ij}	$\{0,1\}$	Kronecker delta
η	$\mathbb{R}_+$	hyperparameter
λ	$\mathbb{R}_+$	hyperparameter
$\boldsymbol{\Lambda}$	$\mathbb{R}^{s\times s}_{\succ 0}$	symmetric positive definite hyperparameter matrix
$\phi(\cdot)$	$\mathbb{R}^d \to \mathbb{R}^{d_f}$	feature map, transforming an input from a d-dimensional space to a d_f-dimensional space
$\phi_c(\cdot)$	$\mathbb{R}^d \to \mathbb{R}^{d_f}$	centered feature map $\phi_c(\cdot) \triangleq \phi(\cdot) - \Sigma_i \phi(x_i)/n$
Φ	$\mathbb{R}^{n\times d_f}$	matrix containing all feature maps as row vectors
Φ_c	$\mathbb{R}^{n\times d_f}$	matrix containing all centered feature maps as row vectors
$\psi(\cdot,\cdot)$	$\mathbb{R}^s \times \{\{\mathbb{R}^s\}\} \to \mathbb{R}^s$	aggregation function
$\boldsymbol{a}_v$	$\mathbb{R}^d$	vector containing the aggregated node features
$\boldsymbol{b}$	$\mathbb{R}^p$	bias vector
$\boldsymbol{c}_i$	$\{-1,1\}^p$	the class encoding of point i
$\boldsymbol{C}$	$\{-1,1\}^{n\times p}$	matrix containing all class encodings as row vectors
d_v	$\mathbb{N}$	node degree
$\boldsymbol{e}_i$	$\mathbb{R}^s$ or $\mathbb{R}^p$	error variable
$\boldsymbol{h}_i$	$\mathbb{R}^s$ or $\mathbb{R}^p$	dual variable/hidden representation (GCKMℓ) or dual variable (Semi-SupRKM)
$\boldsymbol{H}$	$\mathbb{R}^{n\times s}$ or $\mathbb{R}^{n\times p}$	matrix containing all dual vectors as row vectors
$\boldsymbol{I}_n$	$\{0,1\}^{n\times n}$	n by n identity matrix
$k(\cdot,\cdot)$	$\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$	a positive definite kernel function
$\boldsymbol{K}$	$\mathbb{R}^{n\times n}$	kernel matrix $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, or the gram matrix $K_{ij} = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$
l_i	$\{0,1\}$	indicator that is equal to one when the label of i is used for training
$\boldsymbol{L}$	$\{0,1\}^{n\times n}$	diagonal matrix containing all label indicators on the diagonal
$\boldsymbol{M}_c$	$\mathbb{R}^{n\times n}$	centering matrix, used to center the kernel matrix (see (5))
n	$\mathbb{N}$	number of datapoints, equal to $\lvert\mathcal{V}_{\text{tr}}\rvert$ in GCKM
p	$\mathbb{R}$	dimensionality of the final representation (the number of classes in one-vs-all encoding)
r_i	$\mathbb{R}$	a weighted combination of v_i and l_i, see (9) and beyond
$\boldsymbol{R}$	$\mathbb{R}^{n\times n}$	diagonal matrix containing all r variables on the diagonal
s	$\mathbb{R}$	dimensionality of a hidden representation
$\boldsymbol{S}$	$\mathbb{R}^{n\times n}$	see (13)
v_i	$\mathbb{R}$	weighting scalar for the datapoints in Semi-SupRKM (the inverse degree of the kernel matrix)
$\boldsymbol{W}$	$\mathbb{R}^{d_f\times s}$	linear transformation matrix
$\boldsymbol{x}_i$	$\mathbb{R}^d$	the feature vector of datapoint i
y_i	$\mathbb{N}$	class label of datapoint i
$\boldsymbol{Z}$	$\mathbb{R}^{s\times s}$	a matrix containing Lagrange multipliers

B Proofs and derivation of the Graph Convolutional Kernel Machine layer

B.1 Proof of Lemma 4.1

Proof. The stationarity conditions of (3) are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial \boldsymbol{W}} = 0 \implies & \boldsymbol{W} = \frac{1}{\eta}\sum_{v=1}^n \phi_c(\boldsymbol{a}_v)\boldsymbol{h}_v^T \iff \boldsymbol{W} = \frac{1}{\eta}\boldsymbol{\Phi}_c^T\boldsymbol{H} \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_v} = 0 \implies & \boldsymbol{h}_v\boldsymbol{\Lambda} = \boldsymbol{W}^T\phi_c(\boldsymbol{a}_v) \iff \boldsymbol{H}\boldsymbol{\Lambda} = \boldsymbol{\Phi}_c\boldsymbol{W} \end{cases}, \tag{17}$$

where $\boldsymbol{\Phi}_c = [\phi_c(\boldsymbol{a}_1), \ldots, \phi_c(\boldsymbol{a}_n)]^T$. Given that $\boldsymbol{K}_c = \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c^T$, and by substituting the first condition into the second, one can eliminate the primal variable $\boldsymbol{W}$:

$$\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{H} = \boldsymbol{H}\boldsymbol{\Lambda}. \tag{18}$$

Next, the Lagrangian of (4) is:

$$\mathcal{L}(\boldsymbol{H}, \boldsymbol{Z}) = -\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{K}_c\boldsymbol{H}) + \text{Tr}(\boldsymbol{Z}^T(\boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s)),$$

and the KKT conditions are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{H}} = -\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{H} + \boldsymbol{H}(\boldsymbol{Z} + \boldsymbol{Z}^T)/2 = 0 \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{Z}} = \boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s = \boldsymbol{0}_s \end{cases}. \tag{19}$$

By choosing $\boldsymbol{\Lambda} = \tilde{\boldsymbol{Z}} = (\boldsymbol{Z} + \boldsymbol{Z}^T)/2$ we obtain (18) from the first condition, which proves the Lemma.

□

B.2 Proof of Proposition 4.2

Proof. Lets define the columns of $\boldsymbol{H}$ as $\boldsymbol{g}_i = [(\boldsymbol{h}_1)_i, \ldots, (\boldsymbol{h}_n)_i]^T$, and $\boldsymbol{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s] = \boldsymbol{H}$. We can then rewrite (19) in vector notation:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{g}_i} = -\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{g}_i + \sum_{j=1}^{s}\tilde{Z}_{ij}\boldsymbol{g}_j = \boldsymbol{0}_n \quad \forall i = 1 \ldots s \\ \frac{\partial \mathcal{L}}{\partial \tilde{Z}_{ij}} = \boldsymbol{g}_i^T\boldsymbol{g}_j - \delta_{ij} = 0 \quad \forall ij = 1 \ldots s \end{cases}, \tag{20}$$

where δ_{ij} is the Kronecker delta, and derive the second order derivatives for the optimization parameters:

$$\nabla^2_{\boldsymbol{g}_i\boldsymbol{g}_j}\mathcal{L} = -\frac{\delta_{ij}}{\eta}\boldsymbol{K}_c + \tilde{Z}_{ij}\boldsymbol{I}_n. \tag{21}$$

By defining $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_s]$, the second order necessary conditions can be formulated as:

$$\sum_{i=1}^{s}\sum_{j=1}^{s}\boldsymbol{d}_i^T(-\frac{\delta_{ij}}{\eta}\boldsymbol{K}_c + \tilde{Z}_{ij}\boldsymbol{I}_n)\boldsymbol{d}_j = -\frac{1}{\eta}\sum_{i=1}^{s}\boldsymbol{d}_i^T\boldsymbol{K}_c\boldsymbol{d}_i + \sum_{i=1}^{s}\sum_{j=1}^{s}\tilde{Z}_{ij}\boldsymbol{d}_i^T\boldsymbol{d}_j \geq 0$$
$$\forall \boldsymbol{D} \in C(\boldsymbol{G}^*), \tag{22}$$

where $C(\boldsymbol{G}^*) = \{\boldsymbol{D} \in \mathbb{R}^{n\times s\times s} \mid \boldsymbol{D}^T\boldsymbol{G}^* = \boldsymbol{0}_s\}$ is the critical cone at $\boldsymbol{G}^*$. For the critical cone, it can be deduced that the following properties hold:

$$\begin{aligned} \boldsymbol{d}_i^T\boldsymbol{g}_j^* + \boldsymbol{d}_j^T\boldsymbol{g}_i^* = 0 \quad \forall ij = 1, \ldots, s \\ \boldsymbol{d}_i^T\boldsymbol{g}_i^* = 0 \quad \forall i = 1, \ldots, s \\ \boldsymbol{d}_i^T\boldsymbol{d}_j = 0 \quad \forall ij = 1, \ldots, s \quad i \neq j. \end{aligned}$$

Without loss of generality, we further assume $\|\boldsymbol{d}_i\| = 1$. From (20), one can derive $\tilde{Z}_{ij} = \frac{1}{\eta}\boldsymbol{g}_j^{*T}\boldsymbol{K}_c\boldsymbol{g}_i^*$. By substituting this and $\boldsymbol{d}_i^T\boldsymbol{d}_j = 0$ in (22), the second order necessary condition becomes:

$$\sum_{i=1}^{s}\frac{1}{\eta}\boldsymbol{g}_i^{*T}\boldsymbol{K}_c\boldsymbol{g}_i^* \geq \sum_{i=1}^{s}\frac{1}{\eta}\boldsymbol{d}_i^T\boldsymbol{K}_c\boldsymbol{d}_i,$$

or after reworking this algebraically:

$$\text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) \geq \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}), \tag{23}$$

with $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ the eigenvectors of $\boldsymbol{K}_c$ with corresponding eigenvalues $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$. For the case where $\text{span}(\boldsymbol{g}_1^*, \ldots, \boldsymbol{g}_s^*) = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$, the left hand side is maximal:

$$\sum_{i=1}^{s}\lambda_i = \text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) \geq \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}).$$

In worst case for $\boldsymbol{D}$ (i.e., $\text{span}(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_s) = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$), there exists an orthonormal transformation such that $\boldsymbol{G}^* = \boldsymbol{D}\boldsymbol{O}$:

$$\sum_{i=1}^{s}\lambda_i = \text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) = \text{Tr}(\boldsymbol{O}^T\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}\boldsymbol{O}) = \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}).$$

We verified that the second order necessary conditions are satisfied for in the case $\text{span}(\boldsymbol{g}_1^*, \ldots, \boldsymbol{g}_s^*) = \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$. Lets now proceed by assuming that $\text{span}(\boldsymbol{g}_1^*, \ldots, \boldsymbol{g}_s^*) \neq \text{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$. In this case there exists a matrix D such that (23) becomes:

$$\text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) < \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}) \leq \sum_{i=1}^{s}\lambda_i.$$

We thus established that the second order necessary conditions are satisfied if and only if span$(\boldsymbol{g}_1^*, \dots, \boldsymbol{g}_s^*) =$ span$(\boldsymbol{v}_1, \dots, \boldsymbol{v}_s)$. Generally, the second order condition is not sufficient. However, as the feasible set $\{\boldsymbol{H} \in \mathbb{R}^{N\times s} \mid \boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s = \boldsymbol{0}_s\}$ is compact, and the objective function $-\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{K}_c\boldsymbol{H})$ is concave, (4) is guaranteed to have a global minimizer Boyd and Vandenberghe [2004]. Therefore, any $\boldsymbol{H}^*$ such that range$(\boldsymbol{H}^*) = span(\boldsymbol{v}_1 \dots \boldsymbol{v}_s)$ is a global minimizer.

□

B.3 Proof of Lemma 4.5

Proof. We start from the vector formulation of the stationarity conditions (17). From the second condition, we get $\hat{\boldsymbol{e}}_v = \hat{\boldsymbol{h}}_v\boldsymbol{\Lambda} = \boldsymbol{W}^T\phi_c(\hat{\boldsymbol{a}}_v)$. By substituting the first condition into this, we obtain:

$$\hat{\boldsymbol{e}}_v = \hat{\boldsymbol{h}}_v\boldsymbol{\Lambda} = \frac{1}{\eta}\sum_{u\in\mathcal{V}_{\text{tr}}} \boldsymbol{h}_u\phi_c(\boldsymbol{a}_u)^T\phi_c(\hat{\boldsymbol{a}}_v),$$

where $\mathcal{V}_{\text{tr}}$ is the set of nodes that is used for training. By doing this for every node in set $\mathcal{V}$, and writing it in matrix notation, we obtain:

$$\hat{\boldsymbol{E}}_{\mathcal{V}} = \hat{\boldsymbol{H}}_{\mathcal{V}}\boldsymbol{\Lambda} = \frac{1}{\eta}\boldsymbol{K}_c^{\mathcal{V},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}.$$

Or in terms of the uncentered kernel matrix: $\mathcal{V}$, and writing it in matrix notation, we obtain:

$$\hat{\boldsymbol{E}}_{\mathcal{V}} = \hat{\boldsymbol{H}}_{\mathcal{V}}\boldsymbol{\Lambda} = \frac{1}{\eta}\boldsymbol{K}^{\mathcal{V},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}} - \frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}^{\mathcal{V}_{\text{tr}},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta}.$$

□

B.4 Proof of Proposition 4.6

Proof. Since the aggregation function is permutation invariant, the kernel evaluations $k(\boldsymbol{a}_u, \boldsymbol{a}_v)$ are as well. Now, let the permutation $\pi(\cdot)$ be defined by a permutation matrix $\boldsymbol{P}$. When $\mathcal{G}$ gets permuted, the rows of the corresponding kernel matrix $\boldsymbol{K}_{\mathcal{V},\mathcal{V}_{\text{tr}}}$ get permuted accordingly by construction. The first term in (8) thus becomes $\frac{1}{\eta}\boldsymbol{P}\boldsymbol{K}_{\mathcal{V},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}$. The second term is a matrix with constant rows, and is therefore permutation invariant: $\frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}_{\mathcal{V}_{\text{tr}},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta} = \boldsymbol{P}\frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}_{\mathcal{V}_{\text{tr}},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta}$. We thus get $\frac{1}{\eta}\boldsymbol{P}\boldsymbol{K}_{\mathcal{V},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}} - \boldsymbol{P}\frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}_{\mathcal{V}_{\text{tr}},\mathcal{V}_{\text{tr}}}\boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta} = \boldsymbol{P}\hat{\boldsymbol{E}}_{\mathcal{V}}$, which proves the permutation equivariance of the mapping. □

B.5 Proof of Lemma 4.7

Proof. The theoretical results of Xu et al. [2019] showed that a message passing iteration of the form:

$$\boldsymbol{h}_v^{(l)} = f^{(l)}\left((1+\epsilon^{(l)})\cdot\boldsymbol{h}_v^{(l-1)} + \sum_{u\in\mathcal{N}_v}\boldsymbol{h}_u^{(l-1)}\right),$$

where $\epsilon^{(l)}$ can be a fixed or a learnable parameter, is maximally powerful in the class of message passing neural networks and as expressive as the one dimensional Weisfeiler-Lehman graph isomorphism test Weisfeiler and Lehman [1968]; and that this follows from the sum aggregator and the injectiveness of the transformation function $f^{(l)}$. Therefore, when GCKMℓ uses the sum aggregator (7), it satisfies the first condition. When the the feature map is chosen to be an RBF-kernel, the second condition is also satisfied because any feature map of the RBF-kernel is injective (we refer to Proposition 4.54, Lemma 4.55, and Corollary 4.58 in Christmann and Steinwart [2008]). □

Remark B.1. from an empirical perspective, Xu et al. [2019] proposed a multilayer perceptron with at least one hidden layer for the function $f^{(l)}$, motivated by the universal approximator theorem Hornik et al. [1989], Hornik [1991]. Anagolously, it has been established that SVMs using the RBF-kernel are universal approximators Burges [1998], Hammer and Gersmann [2003].

C Proofs and derivation of the Semi-Supervised Restricted Kernel Machine

C.1 Proof of Lemma 4.8

Proof. The stationarity conditions of (11) are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial \boldsymbol{W}} = 0 \Longrightarrow & \boldsymbol{W} = \frac{1}{\eta}\sum_i r_i \phi(\boldsymbol{x}_i)\boldsymbol{h}_i^T \Longleftrightarrow \boldsymbol{W} = \frac{1}{\eta}\boldsymbol{\Phi}^T\boldsymbol{R}\boldsymbol{H} \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_i} = 0 \Longrightarrow & \boldsymbol{h}_i = (\boldsymbol{W}^T\phi(\boldsymbol{x}_i) + \boldsymbol{b}) + r_i - 1\frac{l_i \boldsymbol{c}_i}{\lambda_2} \Longleftrightarrow \boldsymbol{H} = \boldsymbol{\Phi}\boldsymbol{W} + \boldsymbol{1}_n\boldsymbol{b}^T + \frac{\boldsymbol{R}^{-1}\boldsymbol{L}\boldsymbol{C}}{\lambda_2} \\ \frac{\partial \bar{J}}{\partial \boldsymbol{b}} = 0 \Longrightarrow & \sum_i r_i \boldsymbol{h}_i = 0 \Longleftrightarrow \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n = \boldsymbol{0}_p \end{cases}, \tag{24}$$

where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]^T$. Given that $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$, and by substituting the first condition into the second, one can eliminate the primal variable $\boldsymbol{W}$:

$$\begin{cases} \boldsymbol{H} = \frac{1}{\eta}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H} + \boldsymbol{1}_n\boldsymbol{b}^T + \frac{\boldsymbol{R}^{-1}\boldsymbol{L}\boldsymbol{C}}{\lambda_2} \\ \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n = \boldsymbol{0}_p \end{cases}. \tag{25}$$

Next, the Lagrangian of (12) is:

$$\mathcal{L}(\boldsymbol{H}, \boldsymbol{z}) = -\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H}) + \frac{1}{2}\text{Tr}(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{H}) - \frac{1}{\lambda_2}\text{Tr}(\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{C}) - \boldsymbol{z}^T\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n,$$

and the KKT conditions are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{H}} = -\frac{1}{\eta}\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H} + \boldsymbol{R}\boldsymbol{H} - \frac{\boldsymbol{L}\boldsymbol{C}}{\lambda_2} - \boldsymbol{R}\boldsymbol{1}_n\boldsymbol{z}^T = \boldsymbol{0}_{n\times s} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n = \boldsymbol{0}_p \end{cases}.$$

By left multiplying the first condition with $\boldsymbol{R}^{-1}$, and moving all negative terms to the right, one obtains (25) with $\boldsymbol{z} = \boldsymbol{b}$. □

C.2 Derivations of Semi-SupRKM

We next proceed with some derivations of which some results are given in section 4.1.

Eliminating $\boldsymbol{W}$ from the stationarity conditions (24) yields the following linear system:

$$\left[\begin{array}{c|c} \frac{1}{\eta}\boldsymbol{K} - \boldsymbol{R}^{-1} & \boldsymbol{1}_n \\ \hline \boldsymbol{1}_n^T & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{R}\boldsymbol{H} \\ \hline \boldsymbol{b}^T \end{array}\right] = \left[\begin{array}{c} -\frac{1}{\lambda_2}\boldsymbol{R}^{-1}\boldsymbol{L}\boldsymbol{C} \\ \hline \boldsymbol{0}_p^T \end{array}\right].$$

Alternatively, using all stationarity conditions, the expression for the bias term becomes:

$$\boldsymbol{b}^T = -\frac{1}{\boldsymbol{1}_n^T\boldsymbol{R}\boldsymbol{1}_n}(\frac{1}{\eta}\boldsymbol{1}_n^T\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H} + \frac{1}{\lambda_2}\boldsymbol{1}_n^T\boldsymbol{L}\boldsymbol{C}),$$

and eliminating both $\boldsymbol{W}$ and $\boldsymbol{b}$ from the stationarity conditions then yields:

$$(\boldsymbol{I}_n - \frac{1}{\eta}\boldsymbol{R}\boldsymbol{S}\boldsymbol{K})\boldsymbol{R}\boldsymbol{H} = \frac{1}{\lambda_2}\boldsymbol{S}^T\boldsymbol{L}\boldsymbol{C}.$$

From the first stationarity condition, we can derive an expression that can be use for out-of-sample extensions:

$$\hat{\boldsymbol{e}} = \boldsymbol{W}^T\phi(\boldsymbol{x}) + \boldsymbol{b} = \frac{1}{\eta}\sum_i r_i\boldsymbol{h}_i\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) + \boldsymbol{b} = \frac{1}{\eta}\sum_i r_i\boldsymbol{h}_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \boldsymbol{b} \tag{26}$$

Note that the second condition gives $\boldsymbol{e}_i = \boldsymbol{h}_i - \frac{l_i\boldsymbol{c}_i}{r_i\lambda_2}$, which simplifies to $\boldsymbol{e}_i = \boldsymbol{h}_i$ for the unsupervised training points. One can thus directly infer the class label $\hat{\boldsymbol{y}}_v$ from the learned representation by comparing the class codes and select the one with closest Hamming distance to the error variable $\boldsymbol{e}_i$. For one-vs-all encoding, this simply corresponds to selecting the index with the highest value.

For $\eta = 1$, these results are the same as those of Mehrkanoon et al. [2015], where their dual variables $\boldsymbol{\alpha}^{(l)}$ correspond to the columns of $\boldsymbol{R}\boldsymbol{H}$.

D Hyperparameter Search

Regarding hyperparameter selection, we tune the hyperparameters of GCKM by random search. The σ^2 of RBF kernels is tuned between e^{-3} and e^5, the employed degree of the polynomial kernel is $p \in \{1, 2\}$, and the t parameter is chosen between e^{-5} and e^5. The number of components is tuned in $s \in \{16, 32, 64\}$. The $\lambda^{(l)}$, $\eta^{(l)}$ are tuned between e^{-4} and e^4. Regarding the baselines, we tune the hyperparameters of each tested method by grid search in the ranges suggested by the authors in their papers.

E Ablation studies

E.1 Contribution of message passing

To assess the contribution of message passing, we compare the performance of GCKM in the semi-supervised setting with fixed splits on the citation networks with that of two simplified alternatives. Recall that GCKM (Figure 1) consists of two GCKMℓ's and a Semi-SupRKM layer. We compare this to two simplified base models. The first base model is also a GCKM with the same architecture, but the aggregation function is $\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) = \boldsymbol{x}_v$ for both GCKMℓ's. This means that actually no aggregation happens and that we obtain a deepRKM Suykens [2017] consisting of two kernel PCA layers followed by a Semi-SupRKM layer. We will refer to this model as deepRKM. By comparing the performance of deepRKM with GCKM, we can assess the importance of the aggregation function and thus the message passing. The second base model is a shallow Semi-SupRKM (13), where we combine the information of the node features with that of the network structure in the kernel matrix, using a trade-off parameter α: $K_{uv} = \alpha\, k(\boldsymbol{x}_u, \boldsymbol{x}_v) + (1 - \alpha)\, e_{u,v}$, where $e_{u,v}$ is a binary variable that indicates whether or not an edge is present between nodes u and v. By comparing Semi-SupRKM with GCKM, we can assess the importance of the iterative message passing framework of the deep model GCKM, w.r.t. a shallow model using the same information.

For both models, we conduct a hyperparameter search and select the best model based on the validation accuracy. Table 5 shows the result of the experiments. For Actor, we observe that the DeepRKM without the message passing achieves a better result than GCKM. Interestingly, the same observation was made for most of the deep learning models in Table 3. For all other datasets, the performance of these base models is indeed inferior to that of GCKM.

Table 5: Test accuracy (%) of base models for semi-supervised setting with fixed splits

METHOD	CHAM.	SQUI.	TEXAS	CORN.	ACTOR	CORA	CITE.	PUBM.
SEMI-SUPRKM	27.11	27.61	50.29	32.37	26.04	59.90	67.60	75.00
DEEPRKM	33.67	24.09	45.09	32.95	**26.52**	46.10	58.50	67.80
GCKM	**41.16**	**30.10**	**56.07**	**38.73**	26.01	**84.20**	**71.80**	**80.10**

E.2 Analysis of training progress and initialization

Analysis of training objective under random initialization Figure 2 shows the training progress on CiteSeer of a model where the dual variables were initialized randomly. Several observations can be made. First, It should be noted that the Cayley Adam algorithm Li et al. [2019] does not guarantee the exact feasibility of the orthogonality constraints. We thus observe a burn-in phase for a few iterations untill the orthogonality loss $\sum_{l=1}^{2} ||\boldsymbol{H}^{(l)^T} \boldsymbol{H}^{(l)} - \boldsymbol{I}||_F$ is small. After this burn-in, we see an almost monotonically decreasing training objective. Second, there is a fast increase in the training accuracy to 100%, whereas the validation and training performance keep increasing more gently, until they reach a certain level and start decreasing again. A third observation is that the cosine similarity increases only slightly for a few iterations until it keeps decreasing. Finally, we observe a sudden change in behavior around iteration 150, caused by another jump in the orthogonality loss. Overall we conclude that minimizing the objective under the constraints indeed yields improving generalization, up to some point, similarly as in training deep neural networks.

Analysis of initialization significance and validation metrics Figure 3 and 4 show the training progress on CiteSeer and Chameleon respectively of models where the dual variables were initialized by sequentially solving the shallow kernel machines, as described in 4.3. For Figure 3, the same hyperparameters were used as in Figure. 2. By comparing Figure 3 with Figure 2, we observe that the test performance of the model after initialization is already better than that of the model without initialization after training. Further, we see that for CiteSeer, the finetuning increases the validation and test accuracy, as well as the cosine similarity score for a few iterations, after which these metrics decrease again.

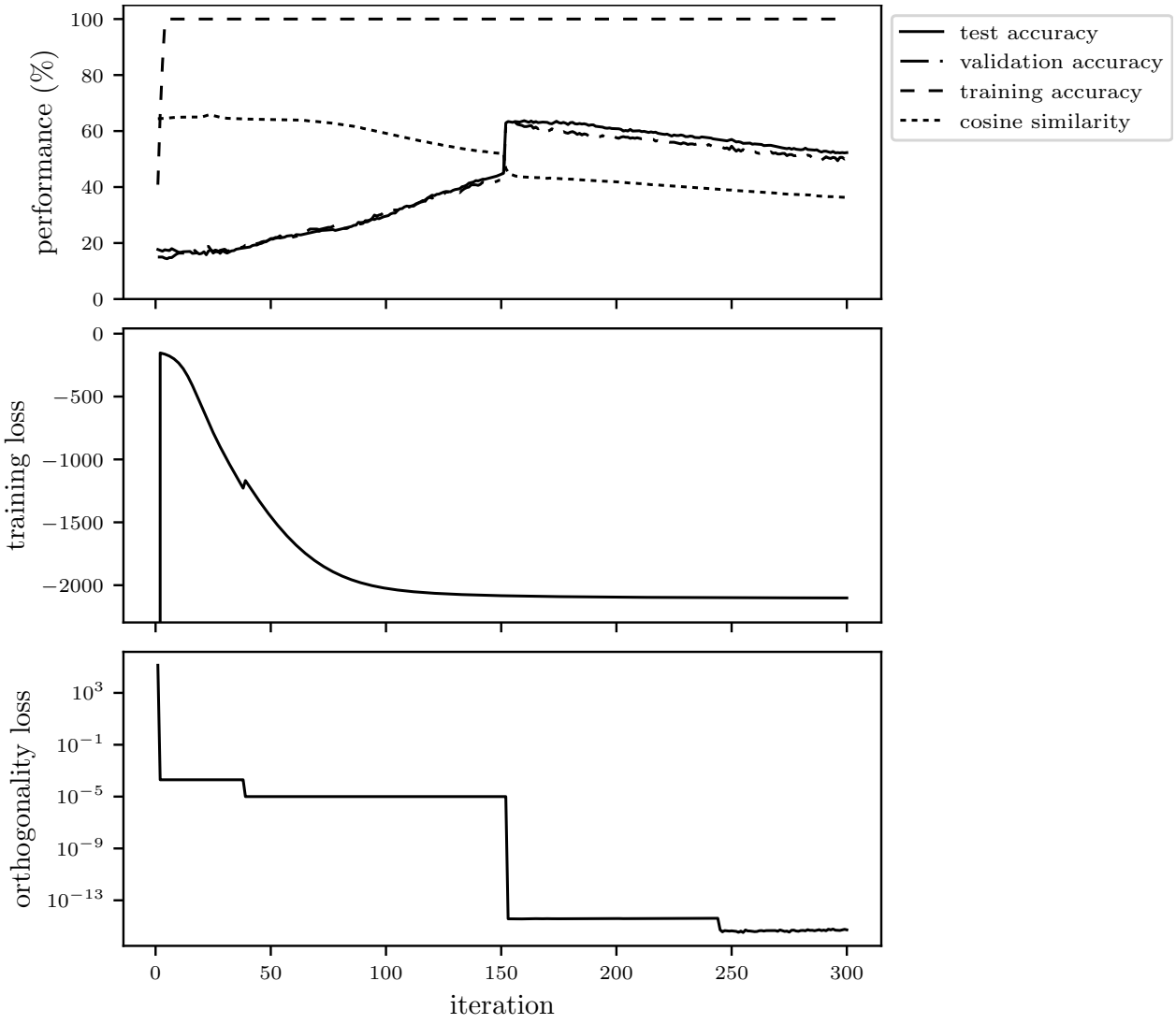


Figure 2: Top: train, validation and test accuracies, and cosine similarity score; middle: training loss; and bottom: orthogonality loss, during training on CiteSeer dataset after random initialization.

For Chameleon, we see a similar trend, though the performance increase is more clear. Generally, the finetuning phase after the sequential initialization only takes few iterations as observed. Next we see that the cosine similarity is also a good indicator for the early stopping of the finetuning phase. For CiteSeer, a homophilious dataset, the trend is indeed well aligned with that of the test accuracy, and although this is less the case for Chameleon, a heterophilious dataset, the general trend is aligned as well. We conclude that the initialization phase is crucial to obtain a good performance, and that the finetuning indeed helps further improving the model at a low cost. We also conclude that both validation accuracy as well as cosine similarity, which is unsupervised, are good indicators for the test performance.

E.3 Computational complexity

The proposed algorithm consists of two main computational parts: eigendecomposition for initialization of the unsupervised blocks and solving a linear system for the semi-supervised block. Initialization of $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ requires computing the first s eigenvectors of the kernel matrices $\boldsymbol{K}_c^{(1)}, \boldsymbol{K}_c^{(2)}$, respectively, with time complexity $\mathcal{O}(sn^2)$. This computation needs to be run only once. Finding $\boldsymbol{H}^{(3)}$ in the employed alternating minimization algorithm requires solving the linear system (13) with worst-case complexity $\mathcal{O}(n^3)$. The emperical running times can be consulted in

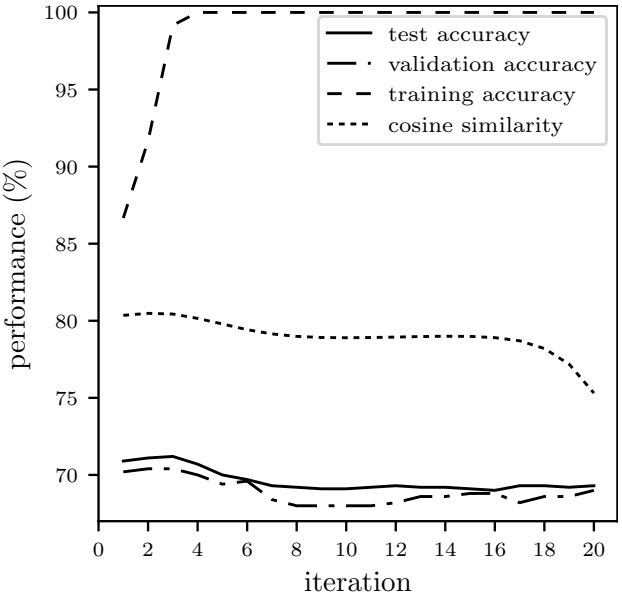


Figure 3: Train, validation and test accuracies, and cosine similarity score during training on CiteSeer dataset.

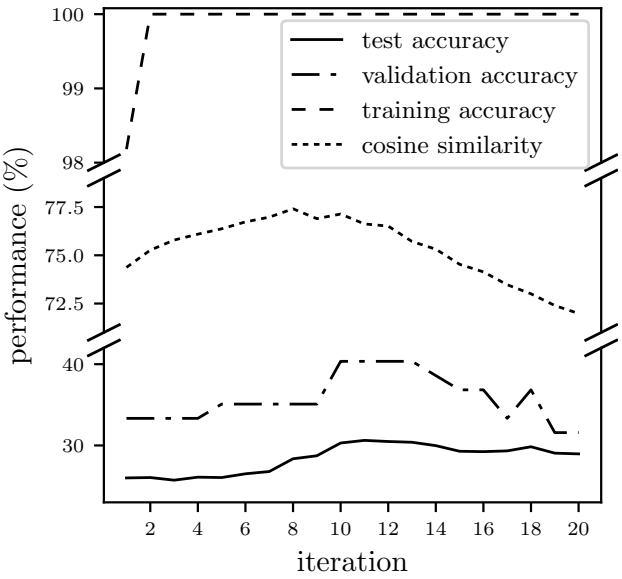


Figure 4: Train, validation and test accuracies, and cosine similarity score during training on Chameleon dataset.

Table 6. When using out-of-sample extensions, one could use a subset of size $m \ll n$, such that the complexity scales w.r.t. m instead of n. First however, a thorough analysis is needed to study the effect of the out-of-sample extension on the hyperparameter choices as well as on the overall performance.

Table 6: Training times for semi-supervised node classification with fixed splits.

METHOD	CHAM.	SQUI.	ACTOR	CORA	CITE.	PUBM.
CHEBNETII	2M7S	1M32S	2M8S	43S	59S	35S
GCKM	1M2S	1M55S	2M49S	29S	49S	3M54S