Title

Exploring the Frontier of Graph-based Approaches for Image and Document Analysis

Author

Chen, Bohan

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Exploring the Frontier of Graph-based Approaches for Image and Document Analysis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Bohan Chen

2024

ABSTRACT OF THE DISSERTATION

Exploring the Frontier of Graph-based Approaches for Image and Document Analysis

by

Bohan Chen
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2024
Professor Andrea L. Bertozzi, Chair

Graph-based machine learning is a powerful framework for analyzing and understanding complex data structures in various domains. This thesis introduces novel graph-based methods in multiple image analysis tasks, including classification, segmentation, and unmixing, as well as their application in enhancing large language models. The key contributions include: (1) the development of new core-set selection and batch active learning methods that significantly improve the efficiency of graph-based active learning while maintaining its effectiveness; (2) the integration of graph learning, active learning, and advanced feature embedding methods to construct pipelines for SAR image classification and multi- or hyperspectral image segmentation, outperforming neural network-based classifiers or segmenters in semi-supervised learning tasks with limited training data; (3) the incorporation of graph-based regularization into the optimization problem of hyperspectral unmixing, enabling the utilization of a small amount of labeled pixels to greatly improve the performance compared to blind unmixing; and (4) the extension of graph Laplacian-based methods to automatically construct knowledge graphs in combination with large language models, enhancing their information retrieval and response generation capabilities.

The proposed methods showcase the effectiveness and versatility of graph-based approaches in addressing challenges such as limited labeled data, computational efficiency, and knowledge representation. The thesis demonstrates the potential of graph-based methods in pushing the boundaries of image and document analysis and their applicability in a wide range of machine learning problems.

The dissertation of Bohan Chen is approved.

Luminita A. Vese

Marcus L. Roper

Stanley J. Osher

Andrea L. Bertozzi, Committee Chair

University of California, Los Angeles

2024

To my advisor, Andrea Bertozzi, thank you for your guidance, insights, and patience. You have not only shaped me as a researcher but also taught me invaluable life lessons.

To all those who have helped and inspired me throughout this doctoral journey. Thank you for being an integral part of my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Andrea Bertozzi. Her guidance, support, and wisdom have been instrumental not only in my academic journey but also in my personal growth. Professor Bertozzi has been an extraordinary mentor, always encouraging me to push the boundaries of my research and think critically. Her passion for science and her dedication to her students have been truly inspiring. Throughout my doctoral studies, Professor Bertozzi has provided me with countless opportunities to collaborate with brilliant researchers, attend conferences, and engage in interdisciplinary projects. Her trust in my abilities and her unwavering support have given me the confidence to tackle challenging problems and explore new ideas. Beyond the realm of research, Professor Bertozzi has also taught me valuable life lessons, such as the importance of perseverance, the power of effective communication, and the value of building strong professional relationships.

I would also like to extend my sincere appreciation to my collaborators, namely, Jocelyn Chanussot, Yifei Lou, Jeff Calder, Jon Schwenk, Kevin Miller, James Chapman, Jason Brown, Harris Hardiman-Mostow, Adrien Weihs, Kaiyan Peng, Tan Zheng, George Mohler, Frederic Schoenberg, Pujan Shrestha, Tara Lyn Slough, and Johannes Urpelainen. Their expertise, insights, and dedication have been instrumental in shaping the research presented in this thesis. I am deeply grateful for the opportunity to work alongside such talented individuals, and I am thankful for the knowledge and skills they have shared with me throughout our collaborations.

I gratefully acknowledge the financial support provided by the UC-National Lab In-Residence Graduate Fellowship Grant L21GF3606 and NSF grant DMS-2027277. These grants support me in conducting the research presented in this thesis, and I deeply appreciate the opportunities and interesting research topics they afforded me.

The UC-National Lab fellowship provided me with the invaluable opportunity to con-

duct research at Los Alamos National Laboratory (LANL), where I was able to broaden my horizons significantly. The experience at LANL allowed me to collaborate with exceptional scientists, access cutting-edge resources, and engage in interdisciplinary research projects. This exposure to a diverse range of scientific perspectives and techniques has greatly enriched my doctoral journey and has had a profound impact on my growth in academia. Furthermore, I would like to thank Dr. Jon Schwenk and his wife at LANL for their support and hospitality during my stay in Los Alamos. Dr. Schwenk kindly offered me the opportunity to rent a room in his home, providing a comfortable and welcoming living environment that greatly enhanced my experience in Los Alamos. The warm and friendly atmosphere, both at Dr. Schwenk's home and throughout the LANL community, created an ideal setting for intellectual growth and personal development.

Chapter 1 provides a high-level overview of the graph-based machine-learning approaches and the structure of this thesis.

Chapter 2 reviews the mathematical background information for this thesis, including graph-related concepts, graph learning, and active learning. It is important to note that this chapter does not contain any novel contributions; rather, it serves as a review of prior work in these areas. The content covered in this chapter forms the mathematical foundation upon which the innovative methods presented throughout the remainder of the thesis are built. By providing a comprehensive overview of the relevant background material, this chapter aims to ensure that readers have the necessary context and understanding to fully appreciate the novel contributions made in the subsequent chapters.

Chapter 3 is related to my papers [CCT23, CMB23a, BCH23]. The paper [CCT23] is approved for public release, NGA-U-2023-00750. The paper [CMB23a] is approved for public release, NGA-U-2023-00757. In papers [CCT23, BCH23], I am the co-first author, and in [CMB23a], I am the sole first author. My main contribution for [CCT23] is proposing the core method, LocalMax batch active learning. In [BCH23], I primarily contribute by proposing the use of graph learning for hyperspectral image segmentation and conducting the

corresponding experiments. For [CMB23a], I apply graph learning and batch active learning for image segmentation and independently complete the main experiments. I acknowledge the contributions to these papers made by the other authors, Andrea Bertozzi, Jocelyn Chanussot, Jeff Calder, Jon Schwenk, Kevin Miller, James Chapman, Jason Brown, Harris Hardiman-Mostow, Adrien Weihs, and Tan Zheng.

Chapter 4 is related to my papers [CMB23b, CMB24]. The paper [CMB23b] was approved for public release, NGA-U-2023-01028. The paper [CMB24] is submitted to the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS). I am the sole first author of these two papers. My contribution to paper [CMB23b] is proposing a graph-based active learning pipeline (GAP) to detect surface water and sediment pixels in multispectral satellite images and conducting extensive experiments. Our GAP requires significantly less data while achieving better results than some neural network-based methods. In [CMB24], my main contribution is improving the GAP by combining it with a feature embedding neural network trained using contrastive learning, which not only further improves accuracy but also greatly enhances efficiency. Additionally, I independently develop a Python-based tool that allows users to easily obtain results from Landsat images for any location on Earth. I acknowledge the contributions to these papers by the other authors, Andrea Bertozzi, Jon Schwenk, and Kevin Miller.

Chapter 5 is related to my paper [CLB23]. I'm the sole first author of this paper. The paper [CMB23b] is approved for public release on May 12, 2023, NGA-U-2023-01028. My main contribution to this paper is proposing a novel hyperspectral unmixing method. I combine the traditional linear mixing model with a graph-based regularization term, allowing the originally unsupervised method to accept ground-truth information for certain pixels. Through detailed experiments, I demonstrate that our new approach achieves an approximately 50% performance improvement over unsupervised methods while only requiring information from about 0.3% of the pixels. I acknowledge the contributions to these papers by the other authors, Andrea Bertozzi, Jocelyn Chanussot, and Yifei Lou.

Chapter 6 is related to my paper [CB23]. I'm the sole first author of this paper. My contribution to this paper is the development of the core method, AutoKG, which enables automatic knowledge graph generation based on language models, as well as a hybrid search method that utilizes these knowledge graphs. Additionally, I created a Python demo that provides a simple test of the methods presented in this paper. I acknowledge the contribution of the only co-author, Andrea Bertozzi, and the assistance of ChatGPT-4 in the first draft of the exposition of the manuscript of this paper.

The author gratefully acknowledges the permissions granted by publishers Springer, IEEE, and SPIE for the reuse of content in this thesis that has been previously published by the author.

VITA

2018	B.S. (Mathematics), Department of Scientific and Engineering Computing, School of Mathematical Science, Peking University; Bachelor of Economics, National School of Development, Peking University.
2018-2021	Graduate Research Assistant, Department of Mathematics, UCLA.
2021-2024	Visiting Researcher, Los Alamos National Laboratory, supported by UC-National Lab In-Residence Graduate Fellowship

PUBLICATIONS

Bohan Chen, and Andrea L. Bertozzi. "AutoKG: Efficient Automated Knowledge Graph Generation for Language Models." In *2023 IEEE International Conference on Big Data (BigData)*, pp. 3117-3126. IEEE, 2023.

Jason Brown, **Bohan Chen**, Harris Hardiman-Mostow, Adrien Weihs, Andrea L. Bertozzi, and Jocelyn Chanussot. "Material Identification in Complex Environments: Neural Network Approaches to Hyperspectral Image Analysis." In *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1-5. IEEE, 2023.

Bohan Chen, Yifei Lou, Andrea L. Bertozzi, and Jocelyn Chanussot. "Graph-Based Active Learning for Nearly Blind Hyperspectral Unmixing" in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1-16, 2023, Art no. 5523716.

Bohan Chen, Kevin Miller, Andrea L. Bertozzi, and Jon Schwenk. "Graph-based Active Learning for Surface Water and Sediment Detection in Multispectral Images", *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, Pasadena, CA, USA, 2023, pp. 5431-5434.

Bohan Chen, Kevin Miller, Andrea L. Bertozzi, and Jon Schwenk. "Batch active learning for multispectral and hyperspectral image segmentation using similarity graphs." *Communications on Applied Mathematics and Computation* (2023): 1-21.

James Chapman, **Bohan Chen**, Zheng Tan, Jeff Calder, Kevin Miller, and Andrea L. Bertozzi. "Novel batch active learning approach and its application on the synthetic aperture radar datasets." In Algorithms for Synthetic Aperture Radar Imagery XXX, vol. 12520, pp. 96-111. SPIE, 2023.

Bohan Chen, Pujan Shrestha, Andrea L. Bertozzi, George Mohler, and Frederic Schoenberg. "A Novel Point Process Model for COVID-19: Multivariate Recursive Hawkes Process." In *Predicting Pandemics in a Globally Connected World, Volume 1: Toward a Multiscale, Multidisciplinary Framework through Modeling and Simulation*, pp. 141-182. Cham: Springer International Publishing, 2022.

Bohan Chen, Kaiyan Peng, Christian Parkinson, Andrea L. Bertozzi, Tara Lyn Slough, and Johannes Urpelainen. "Modeling illegal logging in Brazil." *Research in the Mathematical Sciences* 8, no. 2 (2021): 29.

CHAPTER 1

Introduction

In recent years, graph-based approaches have emerged as a powerful tool for analyzing and understanding complex data structures [Str01, New03], particularly in the domains of image and document analysis [Zhu05, MKB13, JPC21, PLW23]. Graphs provide a natural and intuitive representation of the relationships and dependencies among entities or data points, allowing for the development of sophisticated algorithms that can uncover hidden patterns and extract meaningful insights. This thesis explores the frontier of graph-based approaches, pushing the boundaries of what is possible in image and document analysis by introducing novel techniques and methodologies.

The field of image and document analysis has witnessed significant advancements thanks to the proliferation of deep learning and computer vision techniques [LBD89, HZR16, VSP17]. However, these approaches often rely on large amounts of labeled data, which can be expensive and time-consuming to acquire. Moreover, they may struggle to capture the intricate relationships between objects and regions within an image or document. Graph-based approaches offer a complementary perspective, focusing on the structural and relational aspects of the data. By exploiting the inherent connections and dependencies within the data, graph-based methods can reduce the reliance on extensive labeled datasets while still providing interpretable and robust solutions [Zhu05, WPC20]. This makes graph-based approaches particularly attractive for scenarios where labeled data is scarce or where the underlying structure of the data plays a crucial role in the analysis.

This thesis aims to bridge the gap between graph structure and image or document anal-

ysis, demonstrating the potential of graph-based approaches to tackle challenging problems in these domains. By leveraging the expressive power of graphs, we develop innovative algorithms and frameworks that can efficiently process and analyze complex datasets with limited labeled data. Our contributions advance the state-of-the-art in graph-based methods and provide new insights into the effectiveness of these approaches in various application scenarios.

The remainder of this chapter is organized as follows. Section 1.1 provides a comprehensive review of existing graph-based approaches for image and document analysis, highlighting their strengths and limitations. Section 1.2 presents an overview of the main contributions of this thesis, outlining the key ideas and remarkable experimental results. Finally, Section 1.3 introduces the necessary preliminaries and notations used throughout the thesis.

1.1 A Review of Graph-Related Learning Approaches

Graph learning refers to machine learning with graph structures, in contrast to other approaches such as support vector machines (SVMs) [CV95], random forests (RFs) [Ho95], or multilayer perceptrons (MLPs) [Hay98]. The graph structure encapsulates the macroscopic geometric configurations and the associations between pairs of data within the dataset, offering a unique lens through which to view and analyze data interconnections and dependencies. By leveraging the graph structure, graph learning can effectively capture the complex relationships and dependencies among data points, which is crucial for many real-world applications, especially for the case of limited labeled data. The types of data we commonly encounter in our daily lives, such as sound, text, and images, all have direct mathematical representations. Sound and text can be regarded as time series, while images can be considered as three-dimensional matrices. Graph structures, essentially, are abstract representations of a set of objects and the relationships between them [Wes01]. This structure can express not only the aforementioned types of data, such as sound, text, and images but

also complex structures like social networks and disease transmission.

In the process of machine learning, fully supervised learning utilizes labeled data, while unsupervised learning employs only unlabeled data. Semi-supervised learning, on the other hand, leverages information from both labeled and unlabeled data to make predictions about the unlabeled data [Zhu05]. Here, we focus on graph-based semi-supervised learning, although graph learning also finds applications in unsupervised learning (e.g., Spectral clustering [NJW01, Von07]) and fully supervised learning (e.g., graph neural networks [ZCH20]). In the semi-supervised setting, we consider a graph on the whole available dataset. Graph learning leverages the graph structure to propagate labels from a small set of labeled nodes to a larger set of unlabeled nodes.

There are two common methods for graph-based semi-supervised learning. The first method is Graph Neural Networks (GNNs) [ZCH20], which predict node labels, edge attributes, or entire graph characteristics by learning the complex relationships and patterns between nodes. For instance, Graph Convolutional Networks (GCNs) [KW17, WSZ19, TNX21] leverage convolutional operations on the graph, enabling the model to learn node representations by aggregating features from their neighbors, thus effectively harnessing both graph topology and node features for learning. The second typical graph learning method is the focus of this thesis, which classifies unlabeled nodes in the graph by solving optimization problems related to graph energy. Such optimization problems aim to minimize the graph Dirichlet energy [Eva22] and its variant forms [ZGL03, BLR04], or the Ginzburg–Landau functional [BF12, BM19] under a regularization term based on labeled node information. These methods have a solid mathematical foundation; for instance, Graph Laplace Learning [ZGL03] can be related to Laplace's equation and harmonic functions on graphs. For node classification tasks, the label propagation results derived from these methods offer good interpretability and allow for further analysis, such as uncertainty quantification [BLS18, QSW19].

In the domain of image analysis, graph learning and its related methodologies have seen numerous successful applications. These include noisy image recovery [MKB13, TM13,

VFM20], image or video segmentation [GO09, HLP13, MSB14, GMB14, CBC14, HSB15, BBT18, BPB20], studies using remotely-sensed images to combine LIDAR and optical images [ICB21], image super-resolution [RF17, YRH21], and blind hyperspectral unmixing [QLC19, QLC21].

In the domain of document analysis, graph-related methods are often closely tied to knowledge graphs (KGs) [JPC21, PLW23]. Knowledge graphs have been shown to enhance the reasoning capabilities of language models [XYC19] and reduce hallucinations [JLF23], which are false or nonsensical outputs generated by the models. By integrating structured knowledge from KGs, language models can ground their predictions in real-world facts and relationships, leading to more accurate and coherent results [ZCZ21]. The applications of knowledge graphs in document analysis are extensive, ranging from information extraction [PLW23, ZWL23] and document classification [RRT21] to question answering, text summarization [HWW20], and entity linking [MSP20]. In each of these tasks, KGs can provide valuable semantic information and prior knowledge to guide the systems and improve their performance and interpretability [PRL19, SCM22].

Building upon the concepts of semi-supervised learning with limited labeled data, graph-based active learning takes a step further by actively selecting the most informative data points to be labeled, guided by the graph structure [Set09, Das11]. While semi-supervised learning already reduces the need for labeled data by leveraging unlabeled instances, graph-based active learning can further minimize the labeling requirements or improve the model's performance with the same amount of labeled data [ZLG03, JH12, CZC17, MLB20, MC23].

The goal of active learning is to select the most informative instances for labeling, thereby reducing the labeling effort while maximizing the model's performance [CK13, WIB15]. This is typically achieved through the use of acquisition functions [Set09, JH12, MMS22, MB24], which evaluate the informativeness of unlabeled instances. By leveraging the graph structure, active learning algorithms can identify vertices that are most uncertain [BLS18, BM19, QSW19, MC23], meaning they are difficult to classify confidently, or those that would lead

to the most significant change in the model's predictions if labeled [MMS22, MB24]. By iteratively querying these informative instances and updating the model, active learning can efficiently learn from a small number of labeled examples [WY13]. This selective querying process is particularly valuable in scenarios where labeling data is expensive or time-consuming, as it allows for the efficient allocation of labeling resources.

Moreover, the propagation mechanism in graph-based active learning can propagate label information through the graph, effectively amplifying the impact of each labeled instance and reducing the overall labeling requirements [ZLG03, ZGL03, Zhu05]. In some cases, graph-based active learning has been shown to achieve comparable performance to fully supervised learning with only a fraction of the labeled data [CMB23b, CMB24]. This highlights the potential of graph-based active learning to significantly reduce the burden of data labeling while maintaining high model performance.

This section is not the only review part in this thesis. In each subsequent chapter, namely Chapter 2, Chapter 3, Chapter 4, Chapter 5, and Chapter 6, we will offer more specific reviews relevant to the respective chapter's focus. In Chapter 2, we provide more details about graph learning (Sections 2.1 and 2.2) and active learning (Section 2.3) approaches. In Chapter 3, we will review sequential and batch active learning methods, as well as delve into the topic of Synthetic Aperture Radar (SAR) imagery. Chapter 4 will provide an overview of remote sensing and related image segmentation techniques. In Chapter 5, we will explore hyperspectral imagery and hyperspectral unmixing methods. Finally, Chapter 6 will review knowledge graph approaches in the context of language models.

1.2 Overview and Contributions

In this thesis, our key contributions lie in the development of efficient graph-based batch active learning techniques, the integration of graph learning with advanced feature embedding methods for improved semi-supervised learning performance, the incorporation of graph-

based regularization in hyperspectral unmixing for effective utilization of limited labeled data, and the extension of graph Laplacian-based methods to automatically construct knowledge graphs for enhancing the information retrieval and response generation capabilities of large language models. These contributions collectively advance the frontier of graph-based approaches for image and document analysis, enabling more effective and efficient learning from limited labeled data and unlocking new possibilities for integrating graph learning with other cutting-edge techniques. Our work spans across several domains, as enumerated below:

1. In Chapter 2, we provide a comprehensive review of the mathematical foundations underpinning the novel contributions presented in the following chapters of this thesis. This chapter covers essential graph-related concepts, graph learning, and active learning, which form the basis for the innovative methods introduced in the subsequent chapters. By offering a thorough background, this chapter aims to equip readers with the necessary knowledge to fully appreciate the original work presented in this thesis.

2. In Chapter 3, we focus on batch active learning and propose novel methods to improve the accuracy and efficiency of the learning process. We introduce Dijkstra's Annulus Core-Set (DAC) for core-set generation and LocalMax for batch sampling. Furthermore, we construct comprehensive pipelines that integrate transfer learning-based feature embedding, graph learning, and batch active learning techniques to achieve high-quality and efficient classification of SAR images and segmentation of hyperspectral images with limited labeled data.

3. Chapter 4 presents our work on graph-based active learning for surface water and sediment detection in multispectral images. We propose pipelines that significantly reduce manual labeling costs and improve the accuracy of water and sediment detection. Moreover, we introduce contrastive learning strategies to enhance graph learning efficiency and robustness. Furthermore, we develop an easy-to-use software package to promote the application of our pipelines in global environmental monitoring.

4. In Chapter 5, we address the problem of nearly blind hyperspectral unmixing using graph-based active learning. We propose an effective pipeline to select labeled pixels and apply graph Laplace learning to the hyperspectral unmixing problem, developing the graph learning unmixing (GLU) model. Furthermore, we introduce a novel semi-supervised hyperspectral unmixing model, graph-regularized semi-supervised unmixing (GRSU), which combines graph-based regularization terms with the linear mixing model and a small number of labeled pixels. Our methods, GLU and GRSU, significantly improve HSU performance using only a small number of easily obtainable pseudo labels, bearing practical implications.

5. In Chapter 6, we introduce AutoKG, an innovative method for automated knowledge graph (KG) generation based on a knowledge base comprised of text blocks. AutoKG circumvents the need for training or fine-tuning neural networks, employs pretrained large language models (LLMs) for extracting keywords as nodes, and applies graph Laplace learning to evaluate the edge weights between these keywords. The output is a simplified KG, where edges lack attributes and directionality, possessing only a weight that signifies the relevance between nodes. We also present a hybrid search strategy with prompt engineering, which empowers LLMs to utilize information from the generated KGs effectively.

In summary, this thesis advances the state-of-the-art in graph-based learning and its applications, offering novel methods and pipelines that improve performance, reduce manual labeling costs, and expand the applicability of these techniques across various domains, including image analysis and natural language processing.

1.3 Preliminaries and Notation

Before delving into the main content of this thesis, it is essential to establish some preliminary information and clarify the notation used throughout the document.

Firstly, it is important to note that theorems, lemmas, corollaries, propositions, remarks, definitions, problems, and conjectures share a common counter, ensuring a sequential numbering throughout the thesis. This maintains consistency and helps readers easily navigate and reference these important elements.

Table 1.1 and Table 1.2 provide a comprehensive list of abbreviations utilized throughout this thesis, arranged in alphabetical order for ease of reference. Readers are encouraged to familiarize themselves with these abbreviations to ensure a smooth reading experience and better understanding of the content.

Similarly, Table 1.3 and Table 1.4 offer a comprehensive list of notation used in this thesis, arranged in alphabetical order.

Chapter 2, titled "Background of Graph Learning and Active Learning," serves as the theoretical foundation for the entire thesis. The innovative methods presented in the subsequent chapters heavily rely on the concepts and techniques introduced in this background chapter.

The following chapters, 3, 4, 5, and 6, contain relatively independent content. However, each of these chapters includes its own introduction or background section, as well as a relevant literature review, to provide context and support for the specific topics addressed within them.

Abbreviation	**Description**
ATR	Automatic Target Recognition (Chapter 3)
BA	Boundary Accuracy (Chapter 4)
CGAP	Contrastive Graph Active Learning Pipeline (Chapter 4)
CNN	Convolutional Neural Network [LBD89]
CNNVAE	Convolutional Variational Auto-Encoders [KW13, PGH16]
DAC	Dijkstras Annulus Core-Set (Chapter 3)
DWM	Deep Water Map [IBP19]
EGU	Endmember-Guided Unmixing network [HGY21]
EXT	Exact abundance value (Chapter 5)
FPR	False Positive Rate
GAP	Graph Active Learning Pipeline (Chapter 4)
GCN	Graph Convolutional Network [KW17]
GLNMF	Graph-regularized $\ell_{1/2}$ Nonnegative Matrix Factorization
GLU	Graph Learning Unmixing (Chapter 5)
GNN	Graph Neural Network [ZCH20]
GRSU	Graph-Regularized Semi-supervised Unmixing (Chapter 5)
GT	Ground-truth
GTVMBO	Graph Total Variation Merriman–Bence–Osher scheme [QLC21]
HSI	Hultispectral Image/Imagery/Imaging
HSU	Hyperspectral Unmixing (Chapter 5)
iff	if and only if
KG	Knowledge Graph (Chapter 6)
KNN	K-nearest neighbors [AMN98]
KSC	Kennedy Space Cente dataset
LLM	Large Language Model (Chapter 6)
LocalMax	LocalMax batch active learning approach (Chapter 3)
MC	The model-change acquisition function [MB24] (Eq. (2.57))

Table 1.1: A comprehensive list of abbreviations utilized throughout this thesis, arranged in alphabetical order for ease of reference (Part 1).

Abbreviation	Description
MCVOpt	The combined MC and VOpt acquisition function [MMS22] (Eq. (2.58))
MGR	Multiclass Gaussian Regression graph learning model [MHS15, BHL21]
MLP	Multilayer Perceptron [Hay98]
MSC	Minimum Simplex Convolutional Network [RKS22]
MSI	Multispectral image/imagery/imaging
MSTAR	Moving and Stationary Target Acquisition and Recognition dataset [AD]
NMF	Nonnegative Matrix Factorization [LS99, FLW22]
OA	Overall Accuracy
OH	One-hot pseudo label (Chapter 5)
PDE	Partial Differential Equation
QMV	Quadratic Minimum Volume [LWY12]
RF	Random Forest [Ho95]
RMSE	Root Mean Square Error (Chapter 5)
SAD	Spectral Angle Distance (Chapter 5)
SAR	Synthetic Aperture Radar (Chapter 3)
SimCLR	The self-supervised contrastive learning approach: a Simple framework for Contrastive Learning of visual Representations [CKN20]
SNR	Signal-to-Noise Ratio [Gon09]
SOTA	State-of-the-art
SSL	Semi-supervised Learning
SupCon	Supervised Contrastive learning [KTW20]
SVM	Support Vector Machine [CV95]
TL	Transfer Learning [PY09]
TPR	True Positive Rate
UC	The uncertainty acquisition function [Set09]. If not specified, it refers to the smallest-margin function (Eq. (2.40))
VOpt	The variance optimization acquisition function [JH12] (Eq. (2.51))

Table 1.2: A comprehensive list of abbreviations utilized throughout this thesis, arranged in alphabetical order for ease of reference (Part 2).

Notations	Description
$A. * B$, $A./B$	Component-wise product and division between matrices A and B.
$\mathbf{1}_n$	The all-one vector of dimension n.
A	The matrix of abundance maps (Chapter 5)
$\mathcal{A}$	The acquisition function in the active learning process.
B	The batch size in batch active learning.
d	The dimension of each feature vector $\mathbf{x} \in X$.
D	The degree diagonal matrix of graph G.
$\mathbf{e}_i$	The N-dimensional **column vector** with all zeros except a 1 at the i^{th} entry.
$G = (X, W)$	The graph G with vertices X and the weight matrix W.
k	The margin size of the patch neighborhood in pixel-wise feature extraction (a $(2k+1) \times (2k+1)$ patch centering at the pixel).
K	The number of neighbors in KNN graph construction.
$L, L_{\text{sym}}, L_{\text{rw}}$	The unnormalized, normalized symmetric, and random walk graph Laplacian matrices.
M	The budget of the active learning i.e., maximum size of the labeled set.
n_c	The number of difference classes.
N	The number of elements in the dataset or the number of graph vertices.

Table 1.3: A comprehensive list of notations utilized throughout this thesis, arranged in alphabetical order for ease of reference (Part 1).

Notations	Description
$\mathcal{N}(\mu, \sigma^2)$	A normal distribution with the mean μ and variance σ^2.
$\mathcal{O}$	The asymptotic upper bound or the growth rate of a function.
$\mathcal{P}$	The matrix projection operator (Equation 5.6)
$\mathcal{Q}$	The query set in the active learning process.
r	The rank of the Nyström low-rank approximated matrix.
S	The endmember spectrum matrix (Chapter 5).
$\mathbf{Tr}(\cdot)$	The trace of a matrix.
$\mathbf{u}, \mathbf{u}^*, \mathbf{u}^\dagger$	Graph vertex functions defined on graph vertices. $\mathbf{u}, \mathbf{u}^* : X \to \mathbb{R}^{n_c}$ are the general and the optimal (for graph Laplace learning) vertex functions respectively. $\mathbf{u}^\dagger : X_l \to \mathbb{R}^{n_c}$ maps $\mathbf{x}_i$ of the labeled subset to its one-hot ground-truth label vector.
$U, U^*, U^\dagger, \hat{U}^\dagger$	$U, U^* \in \mathbb{R}^{N \times n_c}, U^\dagger \in \mathbb{R}^{\lvert X_l \rvert \times n_c}$ are the matrix forms of $\mathbf{u}, \mathbf{u}^*, \mathbf{u}^\dagger$ respectively. $\widehat{U}^\dagger \in \mathbb{R}^{N \times n_c}$ is a matrix with the ground-truth one-hot rows $U^\dagger$ corresponding to X_l and zero rows corresponding to X_u.
W	The weight matrix of graph G.
X, X_l, X_u	$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is the whole dataset of N feature vectors in a Euclidean space. X_l is the labeled subset. $X_u = X \setminus X_l$ is the unlabeled subset
X^{mat}	The matrix form of dataset X. (Chapter 5)
$y_i, y_i^\dagger$	The predicted and the ground-truth label index of $\mathbf{x}_i \in X$ respectively.
$\delta(\cdot, \cdot)$	δ-function: $\delta(x, y) = 1$ iff $x = y$, otherwise 0.

Table 1.4: A comprehensive list of notations utilized throughout this thesis, arranged in alphabetical order for ease of reference (Part 2).

CHAPTER 2

Background of the Graph Learning and Active Learning

This chapter reuses materials from the author's publications [CCT23, CMB23a, CLB23]. Content from [CCT23] is reproduced with permission from SPIE, while content from [CMB23a, CLB23] is used under the Creative Commons CC BY license.

This chapter conducts a comprehensive review of several established methodologies related to graph learning that form the fundamental for our innovative approaches introduced in the following chapters. Initially, we delve into the concept of the graph as defined within the realm of discrete mathematics. Following this, we review various graph learning techniques, which classify unlabeled nodes through the minimization of graph-based energy. Lastly, we discuss graph-based active learning strategies, highlighting their significance in iteratively refining model accuracy by selectively querying labels for informative nodes.

2.1 Graph and Related Concepts

In discrete mathematics, a graph is a structure on a set of objects among which certain pairs are interconnected in a specified manner. Specifically, these objects are called "vertices" and there is an "edge" between each related pair of objects. Edges in the graph can be undirected or directed, where one of the two vertices it connects is the starting point and the other is the endpoint. Additionally, vertices and edges can have different attributes, which can be values or vectors, or even more generalized properties defined by natural language (for example,

in knowledge graphs [WMW17]). In more complex scenarios, the graph can evolve into a structure known as a Multigraph. Multigraph allows for multiple edges between a pair of vertices, and edges that connect a vertex to itself.

The thesis uses simple weighted graphs. Simple graphs are characterized by the absence of multiple edges between any two vertices and loops (edges connecting a vertex to itself). In the graph, vertices are d-dimensional vectors and all edges are undirected and each of them has a single positive weight, without any additional attributes.

In the conventional notation, we write $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\} \in \mathbb{R}^d$ is the set of vertices (each of them corresponds to a d-dimensional vector), and $E = \{e_1, e_2, \ldots, e_{N_e}\}$ is the set of edges. Each element $e_i \in E$ is an unordered pair $e_i = (v_{i_1}, v_{i_2})$ corresponding to an edge between vertices v_{i_1}, v_{i_2}. We introduce the concept of an adjacency matrix, denoted by A, of the graph $G = (V, E)$. A is an $N \times N$ matrix where each element A_{ij} represents the presence or absence of an edge between vertices v_i and v_j:

$$A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

In the case of a weighted graph, we can use the weight matrix W as an extension of the adjacency matrix A. Let w_{ij} be the positive weight on the edge between vertices v_i and v_j. Then W is an $N \times N$ matrix given by:

$$W_{ij} = \begin{cases} w_{ij} & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that this weight matrix setting is based on the assumption that all weights w_{ij} are positive.

In the following content, for convenience, we use the graph representation $G = (V, W)$ rather than the standard one $G = (V, E)$. Since the graph G is a simple weighted graph, all the information from the edge set E is fully incorporated into the weight matrix W. This representation $G = (V, W)$ retains all necessary information without loss.

We introduce the graph Laplacian matrix [Mer94]

$$L = D - W, \tag{2.3}$$

where D is a diagonal matrix of diagonal entries $d_1, d_2, \ldots, d_N$ with

$$d_i = \sum_{j=1}^{N} w_{ij}. \tag{2.4}$$

It encodes important geometric information of the graph $G = (V, W)$. There are two widely-used normalized graph Laplacian matrix, the symmetric graph Laplacian L_{sym}, and the random walk graph Laplacian L_{rw}:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}, \tag{2.5}$$

$$L_{\text{rw}} = D^{-1} L = I - D^{-1} W. \tag{2.6}$$

Theorem 1. *[Von07] There are some important properties of the graph Laplacian matrices:*

1. *L and L_{sym} are symmetric and positive semi-definite, and have the quadratic form, i.e. for any $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$:*

$$\mathbf{x}^\top L \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} (x_i - x_j)^2, \tag{2.7}$$

$$\mathbf{x}^\top L_{sym} \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2. \tag{2.8}$$

2. *L, L_{sym}, L_{rw} have the eigenvalue 0. The all-one vector $\mathbf{1} \in \mathbb{R}^N$ is an eigenvector of eigenvalue 0 for L, L_{rw}. $D^{1/2}\mathbf{1}$ is an eigenvector of 0 for L_{sym}.*

3. *All eigenvalues of L, L_{sym}, L_{rw} are non-negative and real-valued.*

4. *Let $G = (V, W)$ be an undirected simple graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of bothL, L_{sym}, L_{rw} equals the number of connected components in the graph. Denote the index set of those k connected components are*

$Z_1, Z_2, \ldots, Z_k \subset \{1, 2, \ldots, N\}$. *Then the eigenspace of* L *and* L_{rw} *is spanned by* $\mathbf{1}_{Z_i}$, $i = 1, 2, \ldots, k$, *where* $\mathbf{1}_Z$ *is a vector with element 1 on indices in* Z *and 0 elsewhere. The eigenspace of* L_{sym} *is spanned by* $D^{1/2}\mathbf{1}_{Z_i}$, $i = 1, 2, \ldots, k$.

Figure 2.1 illustrates the process of deriving the weight matrix W and the Graph Laplacian $L = D - W$ from the structure of a simple undirected weighted graph. This graph includes 5 nodes and 6 edges. The corresponding weight matrix $W \in \mathbb{R}^{5\times5}$ has 12 non-zero elements.

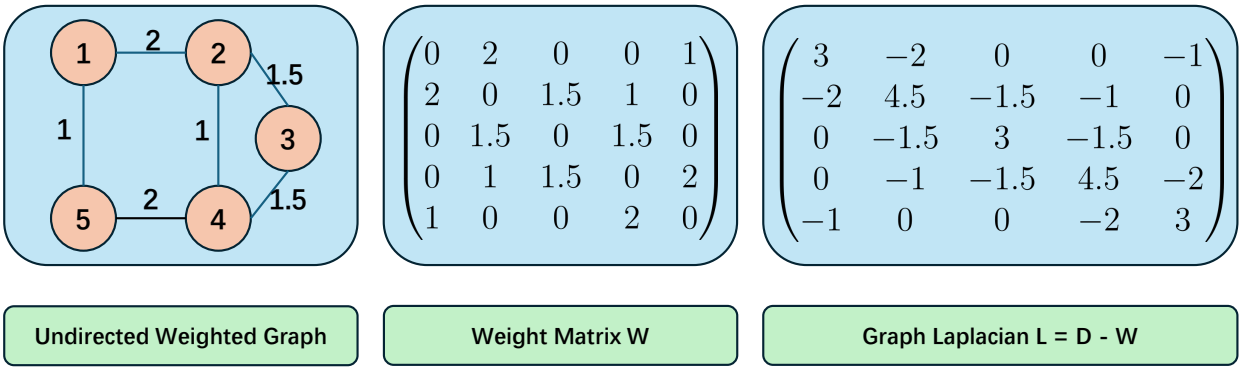


Figure 2.1: An example of the graph Laplacian $L = D - W$.

2.2 Graph Learning

This section introduces in detail how to employ graph learning techniques on a general classification task. Given a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^d$ and a labeled subset $X_l \subset X$, we aim to classify the unlabeled set $X_u = X \setminus X_l$ by graph learning. The general pipeline includes two steps: 1. graph construction; 2. solving a graph-based optimization problem. Specifically, we construct a weighted graph $G = (X, W)$, where the dataset X serves as the set of vertices, and the weights on the edges represent the similarity between the two vertices they connect. The second step is to solve for an optimized node function that minimizes a certain graph energy with a regularization term about the labeled set X_l.

2.2.1 Graph Construction

Based on the dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, we define the graph $G = (X, W)$ with the vertex set X and an edge weight matrix $W \in \mathbb{R}^{N \times N}$. Define $w(\mathbf{x}_i, \mathbf{x}_j)$ as the similarity weight between distinct vertices $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$w(\mathbf{x}_i, \mathbf{x}_j; \tau_i, \tau_j) = \exp\left(-\frac{\angle(\mathbf{x}_i, \mathbf{x}_j)^2}{\sqrt{\tau_i \tau_j}}\right), \tag{2.9}$$

where $\angle(\mathbf{x}_i, \mathbf{x}_j) = \arccos\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)$ computes the angle between feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and τ_i, τ_j are constant parameters related to $\mathbf{x}_i$ and $\mathbf{x}_j$.

If we compute the similarity weight between each pair of the vertices, both the time complexity and space complexity are $\mathcal{O}(N^2)$, which is not acceptable when the number of vertices is large. In practical scenarios, our graphs typically consist of tens of thousands of vertices, necessitating more efficient computational strategies to ensure feasibility and scalability. We introduce two methods to improve computational efficiency, the K-nearest neighbor (KNN) sparse weight matrix, and the Nyström low-rank approximation to the weight matrix.

2.2.1.1 KNN sparse weight matrix

To enhance computational efficiency, we require the $N \times N$ weight matrix W to be sparse. For each vertex $\mathbf{x}_i$, we only consider edges between x_i and its K-nearest neighbors (KNN), based on the angle distance $\angle(\mathbf{x}_i, \mathbf{x}_j)$. This selection process can be efficiently implemented using an approximate nearest neighbor search algorithm [AMN98]. Let $\mathbf{x}_{i_k}$, for $k = 1, 2, \ldots, K$, denote the K-nearest neighbors of $\mathbf{x}_i$ (excluding x_i itself). In addition, the paramter τ_i in the similarity weight function $w(\mathbf{x}_i, \mathbf{x}_j; \tau_i, \tau_j)$ is determined based on the similarity of $\mathbf{x}_i$ to its K^{th} nearest neighbor, specifically, $\tau_i = \angle(\mathbf{x}_i, \mathbf{x}_{i_K})$, with x_{i_K} being the K^{th} nearest neighbor of $\mathbf{x}_i$.

A sparse weight matrix is then defined by

$$\bar{W}_{ij} = \begin{cases} W(\mathbf{x}_i, \mathbf{x}_j), & \text{for } j = i_1, i_2, \ldots, i_K, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

The matrix $\bar{W} = \{\bar{W}_{ij}\}_{i,j=1}^{N}$ may not be symmetric, since in KNN search, x_i can be in the KNN of x_j while x_j is not in the KNN of x_i. The sparse weight matrix $\bar{W}$ is symmetrized to derive the final weight matrix W, redefining $W_{ij} := (\bar{W}_{ij} + \bar{W}_{ji})/2$. The sparseness of W guarantees the sparseness of the graph Laplacian matrices $L, L_{\text{sym}}, L_{\text{rw}}$.

The parameter K for the KNN search is selected to ensure that the constructed graph G remains connected. We can use the Theorem 1 to check the number of connected components in graph G, i.e. check the multiplicity of eigenvalue 0 in any of the graph Laplacian matrices $L, L_{\text{sym}}, L_{\text{rw}}$. The graph is connected iff the multiplicity of eigenvalue 0 is 1.

We begin by setting $K = K_0$ to identify the appropriate K. If the current graph is disconnected, we update K by doubling its value, i.e., $K \leftarrow 2K$. This procedure is repeated until a connected graph is achieved. Practically, selecting $K = 30$ often ensures graph connectivity in most cases.

Remark 2. *Let W be the KNN sparse weight matrix of the dataset X ($|X| = N$). The KNN parameter is K.*

1. *The time complexity for the KNN search is $\mathcal{O}(N \log N)$ [AMN98]. Therefore, the time complexity for the KNN graph construction (i.e. calculating W) is $\mathcal{O}(N \log N)$.*

2. *Both the time and space complexities to calculate $W\mathbf{x}$ for any column vector $\mathbf{x}$ are $\mathcal{O}(KN)$.*

2.2.1.2 Nyström Low-rank Approximation

Another approach to improve the computational efficiency is to use the Nyström extension [FBC04] to have a low-rank approximation of the weight matrix W. Given the rank r, by

randomly selecting a small subset $X_1 \subset X$ with the size $|X_1| = r$, we can partition X into X_1 and $X_2 = X \setminus X_1$. Practically, we choose $r = 100$. We need to fix $\tau_i = \tau$ for $i = 1, 2, \ldots, N$. The dense weight matrix $W_{\text{dense}} = \{w(\mathbf{x}_i, \mathbf{x}_j; \tau, \tau)\}_{i,j=1}^N$ can be written as

$$W_{\text{dense}} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix},$$

where W_{11} denotes the weights of nodes in set X_1, W_{12} denotes the weights between set X_1 and set X_2, $W_{21} = W_{12}^T$ and W_{22} denotes the weights of nodes in set X_2.

The Nyström extension gives an approximation of the dense weight matrix W_{dense} by

$$W_{\text{dense}} \approx W = \begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} W_{11}^{-1} \begin{bmatrix} W_{11} & W_{12} \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{21} W_{11}^{-1} W_{12} \end{bmatrix} \tag{2.11}$$

Let $\mathbf{1}_n$ be the n-dimensional all-one vector, and two column vectors d_1 and d_2 be defined by

$$\begin{aligned} d_1 &= W_{11}\mathbf{1}_r + W_{12}\mathbf{1}_{N-r}, \\ d_2 &= W_{21}\mathbf{1}_r + (W_{21}W_{11}^{-1}W_{12})\mathbf{1}_{N-r}. \end{aligned} \tag{2.12}$$

With equations (2.11) and (2.12), for any vector $\mathbf{x} \in \mathbb{N}$, we can efficiently calculate the product $L\mathbf{x}$ by

$$\begin{aligned} L\mathbf{x} &= \begin{bmatrix} d1. * \mathbf{x}1 \\ d2. * \mathbf{x}2 \end{bmatrix} - \begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} W_{11}^{-1} \begin{bmatrix} W_{11}, W_{12} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \\ &= \begin{bmatrix} d1. * \mathbf{x}1 - W_{11}\mathbf{x}_1 - W_{12}\mathbf{x}_2 \\ d2. * \mathbf{x}_2 - W_{21}\mathbf{x}_1 - W_{21}W_{11}^{-1}W_{12}\mathbf{x}_2 \end{bmatrix}, \end{aligned} \tag{2.13}$$

where $.*$ denotes the component-wise product between two matrices or vectors with the same size.

Similarly, for the normalized symmetric Laplacian matrix $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$, we have:

$$\begin{aligned} L_{\text{sym}}\mathbf{x} &= \left(I - D^{-1/2} \begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} W_{11}^{-1} \begin{bmatrix} W_{11}, W_{12} \end{bmatrix} D^{-1/2} \right) \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{x}1 - D_1^{-1/2} W_{11} D_1^{-1/2} \mathbf{x}_1 - D_1^{-1/2} W_{12} D_2^{-1/2} \mathbf{x}_2 \\ \mathbf{x}_2 - D_2^{-1/2} W_{21} D_1^{-1/2} \mathbf{x}_1 - D_2^{-1/2} W_{21} W_{11}^{-1} W_{12} D_2^{-1/2} \mathbf{x}_2 \end{bmatrix}. \end{aligned} \tag{2.14}$$

Remark 3. *Based on the dataset X ($|X| = N$), we choose a subset X_1 ($|X_1| = r$), and let $X_2 = X \setminus X_1$. Let W be the low-rank approximation matrix of the dense weight matrix W_{dense} by the Nyström extension. Let L and L_{sym} be the corresponding graph Laplacian matrices.*

1. *The Nyström extension requires to calculate the pairwise distance within X_1 and between X_1 and X_2, which has the time complexity $\mathcal{O}(rN)$. In addition, it requires to calculate the inverse W_{11}^{-1}, which requires $\mathcal{O}(r^3)$. The time complexity to construct a graph based on the Nyström extension is $\mathcal{O}(rN + r^3)$.*

2. *To calculate the matrix products $L\mathbf{x}$ and $L_{sym}\mathbf{x}$, it only needs to save matrices W_{11}, W_{21} and W_{11}^{-1}, which takes $\mathcal{O}(r^2 + rN)$ space complexity. According to equations* (2.13) *and* (2.14)*, the time complexity to calculate the products $L\mathbf{x}$ and $L_{sym}\mathbf{x}$ is also $\mathcal{O}(r^2 + rN)$.*

Remark 4. *In this thesis, we do not employ the Nyström extension method for obtaining the eigenvalues and eigenvectors of the kernel matrix W. Instead, we utilize it to procure a low-rank approximation of W and to accelerate the computation of the product of corresponding graph Laplacian matrices, i.e., $L\mathbf{x}$ and $L_{sym}\mathbf{x}$. For the extraction of eigenvalues and their corresponding orthogonal eigenvectors, a more refined approach is required, as discussed in [BF12, MMK17].*

2.2.2 Graph Laplace Learning

Based on the dataset $X \subset \mathbb{R}^d$, we construct a graph $G = (X, W)$ as described in the previous Section 2.2.1. we present previous work on the graph-based approach for semi-supervised learning in this part.

Assume we have the ground truth label information on the dataset $X_l \subset X$. We aim to classify the unlabeled set $X_u = X \setminus X_l$ into n_c classes indexed by $1, 2, \ldots, n_c$. Let $\mathbf{u}^\dagger : X_l \to \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{n_c}\}$ be the ground-truth labeling function that maps each feature vector $\mathbf{x}_i \in X_l$ to a one-hot class label vector $\mathbf{y}_i^\dagger = \mathbf{u}^\dagger(\mathbf{x}_i)$, where $\mathbf{e}_i$ is the i^{th} standard basis vector with all zeros except a 1 at the i^{th} entry. Let $y_i^\dagger$ be the ground-truth label index of the one-hot vector $\mathbf{y}_i^\dagger$, i.e. $y_i^\dagger = \arg\max \mathbf{y}_i^\dagger$.

The inferred classification of unlabeled vertices X_u comes from thresholding a continuous-valued node function $\mathbf{u} : X \to \mathbb{R}^{n_c}$. In particular, the predicted label of $\mathbf{x}_i \in X$ is $y_i = \arg\max\{u_1(\mathbf{x}_i), u_2(\mathbf{x}_i), \ldots, u_{n_c}(\mathbf{x}_i)\}$, where $u_k(\mathbf{x}_i)$ is the k^{th} entry of $\mathbf{u}(i)$. Consider a $N \times n_c$ matrix U, whose i^{th} row is $\mathbf{u}(i)$; that is, each node function $\mathbf{u}$ can be identified by a matrix U whose ith row represents the output of $\mathbf{u}$ at $\mathbf{x}_i$. The graph-based semi-supervised learning (SSL) model that we consider obtains an optimal U^* (i.e. optimal node function $\mathbf{u}^*$) by solving an optimization problem of the form:

$$\begin{aligned} U^* &= \arg\min_{U \in \mathbb{R}^{N \times n_c}} \frac{1}{2}\langle U, LU\rangle_F + \sum_{\mathbf{x}_i \in X_l} \ell(\mathbf{u}(\mathbf{x}_i), \mathbf{u}^\dagger(\mathbf{x}_i)) \\ &= \arg\min_{\mathbf{u}: X \to \mathbb{R}^{n_c}} \mathcal{J}(\mathbf{u}, \mathbf{u}^\dagger) = \arg\min_{U \in \mathbb{R}^{N \times n_c}} \mathcal{J}(U, U^\dagger), \end{aligned} \tag{2.15}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product for matrices, and $U^\dagger \in \mathbb{R}^{|X_l| \times n_c}$ is the matrix of the ground-truth one-hot labels.

The first term in (2.15) is called the graph Dirichlet energy [Eva22]. According to Theorem 1, we have:

$$\frac{1}{2}\langle U, LU\rangle_F = \frac{1}{4} \sum_{\mathbf{x}_i, \mathbf{x}_j \in X} W_{ij} \|\mathbf{u}(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_j)\|^2. \tag{2.16}$$

Note that W_{ij} measures the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. In the minimizing process, the term $W_{ij}\|\mathbf{u}(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_j)\|^2$ ensures that $\mathbf{u}(\mathbf{x}_i)$ and $\mathbf{u}(\mathbf{x}_j)$ are relatively close for similar $\mathbf{x}_i$ and $\mathbf{x}_j$.

The second term of (2.15), $\sum_{\mathbf{x}_i \in X_l} \ell(\mathbf{u}(\mathbf{x}_i), \mathbf{u}^\dagger(\mathbf{x}_i))$, is a regularization term that ensures that the output of $\mathbf{u}$ at the labeled data $\mathbf{x}_i \in X_l$ stays close to the ground-truth $\mathbf{u}^\dagger(\mathbf{x}_i)$.

The function $\ell : \mathbb{R}^{n_c} \times \mathbb{R}^{n_c} \to \mathbb{R}$ measures the difference between the prediction $\mathbf{u}(\mathbf{x}_i)$ and the ground-truth $\mathbf{u}^\dagger(\mathbf{x}_i)$. The SSL scheme introduced in [ZGL03], referred to as Laplace learning, uses the hard-constraint regularization:

$$\ell_h(x, y) = \begin{cases} +\infty, \text{ if } x \neq y, \\ 0, \text{ if } x = y. \end{cases} \tag{2.17}$$

This hard-constraint regularization function ℓ_h forces the minimizer $\mathbf{u}^*$ to be the same as the ground truth $\mathbf{u}^\dagger$ on the labeled set X_l.

We take the following steps to solve the optimization problem (2.15) with the hard constraint (2.17). We can reorder the vertices to be able to write $U = \begin{bmatrix} U_l \\ U_u \end{bmatrix}$, where U_l corresponds to the submatrix of U whose rows correspond to the labeled set X_l and U_u similarly corresponds to the unlabeled set X_u. Likewise, we can split the weight matrix W, degree matrix D, and Laplacian matrix into labeled and unlabeled submatrices as

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}, \quad D = \begin{bmatrix} D_{ll} & D_{lu} \\ D_{ul} & D_{uu} \end{bmatrix}, \quad L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}. \tag{2.18}$$

As a result of the hard-constraint labeling of Laplace learning, U_l^* is fixed as the one-hot encodings of the ground-truth labels on the labeled set X_l; that is

$$U_l^* = \begin{bmatrix} \mathbf{u}^\dagger(\mathbf{x}_{i_1}) \\ \mathbf{u}^\dagger(\mathbf{x}_{i_2}) \\ \vdots \\ \mathbf{u}^\dagger(\mathbf{x}_{i_{|X_l|}}) \end{bmatrix}, \quad X_l = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_{|X_l|}}\}.$$

According to [ZGL03], the optimizer U_u^* of Laplace learning can be calculated explicitly as

$$U_u^* = (D_{uu} - W_{uu})^{-1} W_{ul} U_l^* = -L_{uu}^{-1} L_{ul} U_l^*. \tag{2.19}$$

This solution (2.19) is derived from the harmonic property of the optimal function $\mathbf{u}^*$. We define the graph Laplacian operator $\mathcal{L}$.

Definition 5. *On graph $G = (X, W)$, let $\mathcal{F} = \{\mathbf{u} : X \to \mathbb{R}^d\}$ be the space of the graph node functions. The graph Laplacian operator $\mathcal{L} : \mathcal{F} \to \mathcal{F}$ is a mapping within $\mathcal{F}$ defined by:*

$$[\mathcal{L}\mathbf{u}](\mathbf{x}_j) = \sum_{\mathbf{x}_i \in X} W_{ij}(\mathbf{u}(\mathbf{x}_j) - \mathbf{u}(\mathbf{x}_i)). \tag{2.20}$$

To derive the formula (2.19), we take the partial derivative of $\mathcal{J}(U, U^\dagger)$ about $\mathbf{u}(\mathbf{x}_j))$ for a single unlabeled vertex $\mathbf{x}_j \in X_u$ and set the derivative to be 0:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \mathbf{u}(\mathbf{x}_j))} &= [\mathcal{L}\mathbf{u}](\mathbf{x}_j) = \sum_{\mathbf{x}_i \in X} W_{ij}(\mathbf{u}(\mathbf{x}_j) - \mathbf{u}(\mathbf{x}_i)) = 0 \\ \Longrightarrow \mathbf{u}(\mathbf{x}_j) &= \frac{1}{d_j} \sum_{\mathbf{x}_i \in X} W_{ij}\mathbf{u}(\mathbf{x}_i). \end{aligned} \tag{2.21}$$

Equation (2.21) reveals the harmonic property of the optimal graph function $\mathbf{u}^*$. Considering all $\mathbf{x}_j \in X_u$ gives the matrix form solution (2.19).

Remark 6. *Practically, when we apply the formula* (2.19) *to solve for the optimal matrix U_u^*, we do not calculate the inverse matrix L_{uu}^{-1}. We use the preconditioned conjugate gradient method to solve the equation:*

$$L_{uu}U_u^* = -L_{ul}U_l^*, \tag{2.22}$$

with the preconditioning matrix

$$P_{CG} = diag\left(\frac{1}{l_1}, \frac{1}{l_2}, \ldots, \frac{1}{l_{|X_u|}}\right), \tag{2.23}$$

where $l_1, l_2, \ldots, l_{|X_u|}$ are diagonal entries of the matrix L_{uu}.

According to Remarks 2 and 3, the product $L_{uu}\mathbf{x}$ can be efficiently calculated by either the KNN sparse or the Nyström approximation. The time and space complexities of calculating such a product are $\mathcal{O}(N)$ when the KNN parameter $K \ll N$ or the Nyström rank parameter $r \ll N$. This is much more efficient than calculating the inverse matrix with a $\mathcal{O}(N^3)$ complexity, in the case that the unlabeled set X_u constitutes the vast majority of the dataset X.

If the graph Laplacian in the problem (2.15) is the normalized L_{sym}, formula (2.19) also works by replacing L_{uu} and L_{ul} with the corresponding block sub-matrices of the L_{sym}.

There are some extended graph SSL schemes based on the optimization problem (2.15). The main difference between them and the Laplace learning scheme is the choice of regularization function ℓ. The multiclass Gaussian regression (MGR) model [MHS15, BHL21] applies a L_2-norm regularization function $\ell_\gamma(x, y) = \frac{1}{2\gamma^2}\|x - y\|_2^2$. The cross-entropy model [JZL19, KW17, MB24] applies a cross-entropy regularization function $\ell_{\text{ce}}(x, y) = -\sum_{m=1}^{n_c} x_m \ln(y_m)$, where $x = (x_1, x_2, \ldots, x_{n_c})$ and $y = (y_1, y_2, \ldots, y_{n_c})$ are properly normalized to lie on the $(n_c - 1)$-simplex.

2.2.3 Extended Schemes for Low Label Rates

The primary graph learning methodology applied in this thesis is the graph Laplacian learning mentioned in the previous section (Section 2.2.2). This part introduces several extended graph learning methods, to provide readers with a more comprehensive understanding of the field.

In SSL, when the label rate, $|X_l|/|X|$, is low, the graph Laplace learning method is not well-posed and can make poor predictions on the unlabeled set X_u [NSZ09, ECR16]. The ***p*-Laplace learning** [ZS05, ECR16] has been developed to address the ill-posedness. For a positive integer $p \geq 2$, the Laplace learning optimization problem (2.15) is modified into:

$$\begin{aligned} \mathbf{u}_p^* = \underset{\mathbf{u}: X \to \mathbb{R}^{n_c}}{\arg\min} \left(\sum_{\mathbf{x}_i, \mathbf{x}_j \in X} W_{ij}^p \|\mathbf{u}(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_j)\|^p \right)^{\frac{1}{p}} := \left(\mathcal{J}_p(\mathbf{u})\right)^{\frac{1}{p}}, \\ \text{s.t.} \quad \mathbf{u}(\mathbf{x}_i) = \mathbf{u}^\dagger(\mathbf{x}_i), \quad \forall \mathbf{x}_i \in X_l. \end{aligned} \tag{2.24}$$

It is shown that the p-Laplace learning performs better than the original Laplace learning when $p > 2$ and the label rate is very low [Cal18, FCL22]. The **Lipschitz learning** [KRS15, Cal19] employs the case $p = +\infty$ for the problem (2.24). The Lipschitz learning solves the

problem:

$$\mathbf{u}_\infty^* = \underset{\mathbf{u}:X\to\mathbb{R}^{n_c}}{\arg\min}\left(\max_{\mathbf{x}_i,\mathbf{x}_j\in X} W_{ij}\|\mathbf{u}(\mathbf{x}_i)-\mathbf{u}(\mathbf{x}_j)\|_1\right), \quad \text{s.t.} \quad \mathbf{u}(\mathbf{x}_i)=\mathbf{u}^\dagger(\mathbf{x}_i), \quad \forall \mathbf{x}_i\in X_l. \tag{2.25}$$

The optimizer $\mathbf{u}_\infty^*$ is the limit of the (unique) minimizers of $\mathbf{u}_p^*$ as p grows to infinity [EH90].

Another extension of the Laplace learning designed for low label rates is the **Poisson learning** [CCT20]. Recall that the graph Laplace learning can be written in the form:

$$[\mathcal{L}\mathbf{u}](\mathbf{x}_j) = 0, \qquad \mathbf{x}_j\in X_u, \tag{2.26}$$

$$\mathbf{u}(\mathbf{x}_i) = \mathbf{u}^\dagger(\mathbf{x}_i), \qquad \mathbf{x}_i\in X_l. \tag{2.27}$$

where $\mathcal{L}$ is the graph Laplacian operator in Definition 5. Define the average ground-truth label by:

$$\bar{\mathbf{u}}^\dagger = \frac{\sum_{\mathbf{x}_i\in X_l}\mathbf{u}^\dagger(\mathbf{x}_i)}{|X_l|}. \tag{2.28}$$

Poisson learning keeps the harmonic property (2.26) on the unlabeled set X_u and modifies the hard constraint (2.27) in Laplace learning:

$$[\mathcal{L}\mathbf{u}](\mathbf{x}_j) = 0, \qquad \mathbf{x}_j\in X_u, \tag{2.29}$$

$$[\mathcal{L}\mathbf{u}](\mathbf{x}_i) = \mathbf{u}^\dagger(\mathbf{x}_i)-\bar{\mathbf{u}}^\dagger, \qquad \mathbf{x}_i\in X_l, \tag{2.30}$$

$$\sum_{j=1}^N d_j\mathbf{u}(\mathbf{x}_j) = 0, \tag{2.31}$$

where equation (2.31) is a regularization for the Poisson learning to have a unique solution and d_i is the degree (2.4) of the vertex $\mathbf{x}_i$.

Another direction to avoid the degeneration of the Laplace learning solution with a low label rate is to adjust the weight matrix, referred as to re-weighting [CS20, MC23]. One approach is to modify the weight W_{ij} based on the distance between $\mathbf{x}_i$ and the labeled set X_l [CS20]:

$$\widetilde{W}_{ij} = \gamma(\mathbf{x}_i)W_{ij}, \text{ where } \gamma(\mathbf{x}_i) = \operatorname{dist}(\mathbf{x}_i, X_l)^{-\alpha}, \ \alpha > d-2. \tag{2.32}$$

This re-weighting approach is extended by solving the graph Poisson equation [MC23]:

$$\begin{aligned} \widetilde{W}_{ij} &= \gamma(\mathbf{x}_i)\gamma(\mathbf{x}_j)W_{ij}, \\ \sum_{\mathbf{x}_k \in X} W_{jk}(\gamma(\mathbf{x}_j) - \gamma(\mathbf{x}_k)) &= \sum_{\mathbf{x}_i \in X_l} \left(\delta_{ij} - \frac{1}{N}\right), \ \forall \mathbf{x}_j \in X. \end{aligned} \tag{2.33}$$

This re-weighted matrix $\widetilde{W} = \{\widetilde{W}_{ij}\}_{i,j=1}^{N}$ is used to replace the weight matrix W in the graph Laplace learning optimization problem (2.15).

2.3 Graph-based Active Learning

Graph-based active learning is a technique used in semi-supervised learning scenarios where the goal is to select the most informative instances from a pool of unlabeled data to be labeled by an oracle (e.g., a human expert) [Set09, Das11]. This approach leverages the underlying structure of the data, which is represented as a graph, to identify the instances that are most useful for improving the performance of the learning model [MMS22, MB24]. By intelligently selecting the most informative instances for labeling, graph-based active learning maximizes the performance of the machine learning model while minimizing the labeling effort.

Active learning is an iterative process that selects a *query set* $\mathcal{Q}$ to augment the current labeled set $X_l \leftarrow X_l \cup \mathcal{Q}$. The general pipeline for graph-based active learning can is illustrated in Figure 2.2, and described as Algorithm 1. The core of active learning is the *Acquisition Function* $\mathcal{A} : X_u \to \mathbb{R}$ which evaluates the informativeness of each unlabeled instance $x \in X_u$. The acquisition function assigns a score to each unlabeled instance, indicating its potential usefulness for improving the model's performance.

Algorithm 1 Graph-based Active Learning: A General Pipeline

Require: The whole dataset X, initial labeled subset X_l, labeling budget M

Ensure: Updated labeled set X_l

1: **Graph Construction:** Construct a similarity graph $G = (X, W)$ with nodes X according to Section 2.2.1
2: **while** $|X_l| < M$ **do**
3: **Prediction:** Based on the current labeled set X_l, predict labels on the unlabeled set $X_u = X \setminus X_l$
4: **Acquisition Function Values:** Calculate the acquisition function values $\mathcal{A}(\mathbf{x}_j)$ of each $\mathbf{x}_j \in X_u$
5: **Query Set Selection:** Select a query set $\mathcal{Q} \subset X_u$ based on the graph structure G and the current acquisition function values $\{\mathcal{A}(\mathbf{x}_j) | \mathbf{x}_j \in X_u\}$
6: Update the labeled set $X_l \leftarrow X_l \cup \mathcal{Q}$
7: Update the unlabeled set $X_u \leftarrow X_u \setminus \mathcal{Q}$
8: **end while**

In Algorithm 1, the graph construction is detailed in Section 2.2.1 and the prediction is based on the graph Laplace learning method (Section 2.2.2). We will further discuss different choices of the acquisition functions and the query set selection approaches in this section.

2.3.1 Bayesian Interpretation and Low-Rank Covariance Matrix

Before discussing different acquisition functions, we provide another perspective on the graph-based SSL models introduced in Section 2.2.2. This part serves as a preparation to help us better understand the mathematics behind those acquisition functions.

A Bayesian interpretation of graph-based SSL models of the form $\mathcal{J}(U, U^\dagger)$ as in (2.15) provides further insight into the confidence of inferred classification on the unlabeled nodes [BLS18, QSW19, MLB20]. The minimizer in (2.15) is equivalent to the *maximum a posteriori*

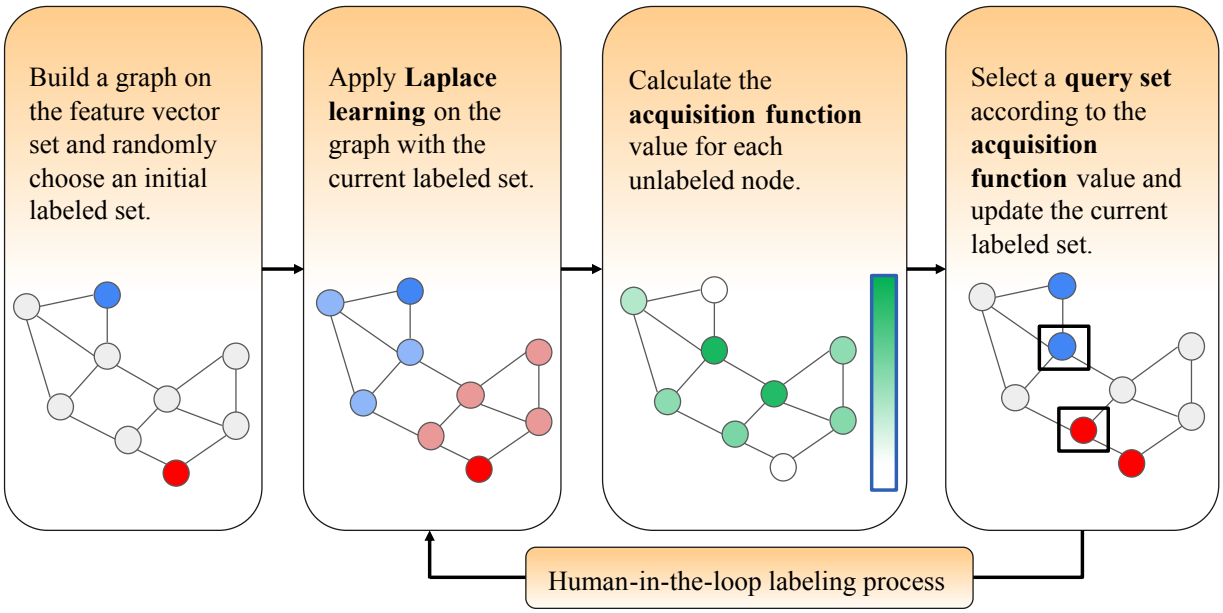


Figure 2.2: Flowchart of the active learning process. The active learning loop is based on a fixed graph. In each step, we apply Laplace learning on the graph and update the labeled set with a query set selected based on the current acquisition function values. It should be noticed that it might need the human-in-the-loop process to obtain the label of the selected query set in each step of the active learning process.

(MAP) estimate of a posterior probability distribution with probability density function:

$$\begin{aligned}\mathbb{P}(U|\mathbf{u}^\dagger) &\propto \exp\left(-\mathcal{J}(U,U^\dagger)\right)\\ &= \exp\left(-\frac{1}{2}\langle U, LU\rangle_F\right)\exp\left(-\sum_{\mathbf{x}_i\in X_l}\ell(\mathbf{u}(\mathbf{x}_i),\mathbf{u}^\dagger(\mathbf{x}_i))\right)\\ &\propto \mu(U)\exp\left(-\Phi_\ell(U,U^\dagger)\right),\end{aligned} \tag{2.34}$$

where $\mu(U)$ can be interpreted as a Gaussian prior on U with a covariance matrix related to the graph Laplacian L and the likelihood $q(U,U^\dagger) \propto \exp\left(-\Phi_\ell(U,U^\dagger)\right)$ related to the functional $\Phi_\ell(U,U^\dagger) = \sum_{\mathbf{x}_i\in X_l}\ell(\mathbf{u}(\mathbf{x}_i),\mathbf{u}^\dagger(\mathbf{x}_i))$. The resulting form of the posterior $\mathbb{P}(U|U^\dagger)$ depends on the choice of loss function ℓ.

For the graph Laplace learning, since the hard constraint 2.17 is applied, we need to consider the distribution $U_u|U^\dagger$ rather than $U|U^\dagger$. Solving for the optimal U_u^* associates to the Gaussian Random Field (GRF) on the graph G [ZLG03], which gives the conditional Gaussian distribution:

$$U_u|U^\dagger \sim \mathcal{N}(U_u^*, L_{uu}^{-1}), \tag{2.35}$$

where U_u^* is the solution to the graph Laplace learning problem given by the equation (2.19) and L^{uu} is the block submatrix of the graph Laplacian matrix L corresponding to the unlabeled set X_u. Note that the distribution (2.35) is about a matrix in $\mathbb{R}^{|X_u|\times n_c}$, which should have a 4-dimensional tensor as the covariance matrix. Writing it as L_{uu}^{-1} means that each column of U_u shares the same covariance matrix L_{uu}^{-1}.

When the MGR regularization function is applied, i.e. $\ell(x,y) = \ell_\gamma(x,y) = \frac{1}{2\gamma^2}\|x-y\|_2^2$, $\mathbb{P}(U|U^\dagger)$ is a Gaussian distribution. The optimal solution to the MGR can be derived by taking the derivative of the matrix U:

$$U_{\mathrm{MGR}}^* = \frac{1}{\gamma^2}\left(L + \frac{1}{\gamma^2}P^\top P\right)^{-1}\widehat{U}^\dagger =: \frac{1}{\gamma^2}C_{\mathrm{MGR}}\widehat{U}^\dagger, \tag{2.36}$$

where $P \in \mathbb{R}^{|X_l|\times N}$ is the projection matrix from the full indices $\{1,2,\ldots,N\}$ onto the indices of the labeled subset X_l, and $\widehat{U}^\dagger \in \mathbb{R}^{N\times n_c}$ is a matrix with the ground-truth one-hot rows corresponding to X_l and zero rows corresponding to X_u. The corresponding conditioned Gaussian distribution is:

$$U|U^\dagger \sim \mathcal{N}(U_{\mathrm{MGR}}^*, C_{\mathrm{MGR}}), \tag{2.37}$$

where each column of U shares the same covariance matrix C_{MGR}.

To efficiently calculate the matrix C_{MGR} since it requires to calculate the inverse matrix, we consider the low-rank approximation of the graph Laplacian L. Since the graph G is connected, the corresponding Laplacian matrix L has exactly one zero eigenvalue. We may order the eigenvalues of L as $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_N$, and then consider the smallest $n \ll N$ eigenvalues. $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries $\lambda_1, \lambda_2, \ldots, \lambda_n$

and $V = [\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^m] \in \mathbb{R}^{N\times n}$ is the matrix of corresponding eigenvectors. $\mathbf{v}^i$ is the eigenvector of eigenvalue λ_i.

With the matrices V, Λ, we can approximate the covariance matrix C_{MGR} by:

$$C_{\text{MGR}} \approx V\left(\Lambda + \frac{1}{\gamma^2}V^\top P^\top PV\right)^{-1} V^\top := V\Sigma_{\text{MGR}}V^\top. \tag{2.38}$$

Since both matrices Λ and $V^\top P^\top PV$ are semi-positive definite, it needs to be proved that the matrix $C_0 := \Lambda + \frac{1}{\gamma^2}V^\top P^\top PV$ is invertible.

Theorem 7. *For any $n \le N$, $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $V = [v^1, v^2, \ldots, v^n]$, the matrix*

$$C_0 := \Lambda + V^\top\left(\frac{1}{\gamma^2}P^\top P\right)V \tag{2.39}$$

is invertible if the graph G is connected.

Proof. As referred in Theorem 1, when the graph G is connected, its graph Laplacian matrix L has exactly one zero eigenvalue $\lambda_1 = 0$ with corresponding eigenvector $\mathbf{v}^1 = (1, 1, \ldots, 1)^\top \in \mathbb{R}^N$.

Since $V^\top P^\top PV$ and Λ are semi-positive definite matrices, we have $x^\top C_0 x \ge 0$ for any column vector $\mathbf{x} \in \mathbb{R}^n$. Consider $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n$ such that $x^\top C_0 x = 0$, which implies $x^\top(V^\top P^\top PV)x = x^\top \Lambda x = 0$. Because $\lambda_2, \ldots, \lambda_n > 0$, we have $x_2 = x_3 = \ldots = x_M = 0$.

Let $\mathbf{v}^j = (v_1^j, v_2^j, \ldots, v_N^j)^\top$, $j = 1, 2, \ldots, M$. Since P is a projection matrix, assume the first row of P is $\mathbf{e}_l$, the l^{th} standard basis vector in $\mathbb{R}^N$. Recall that $v_1^1 = v_2^1 = \ldots = v_N^1 = 1$, we have

$$0 = x^\top(V^\top P^\top PV)x = x_1^2\sum_{j=1}^{M}(v_l^j)^2 \ge x_1^2 v_l^1 = x_1^2.$$

Hence $x_1 = 0$, x is a zero vector. This implies that C_0 is a positive definite matrix, and of course, invertible. □

2.3.2 Acquisition Functions

As mentioned before, the acquisition function quantifies the benefit of acquiring the ground-truth label of each unlabeled data to the model's performance. Since the focus in this thesis is the graph-based learning approaches, the acquisition functions we introduce here are designed for the graph learning classifier, including the Uncertainty (UC) [BLS18, MLB20, QSW19], Model-Change (MC) [MLB20, MB24], Variance Optimization (VOpt) [JH12], and Model-Change Variance Optimal (MCVOpt) acquisition functions [MMS22].

2.3.2.1 Uncertainty (UC) Acquisition Functions

The UC acquisition function $\mathcal{A}_{\mathrm{UC}}$ quantifies the uncertainty of the classifier $\mathbf{u}$ on each unlabeled vertex $\mathbf{x}_j \in X_u$ according to the classifier's output. Uncertainty sampling thus prioritizes querying points that are close to the current classifier's decision boundaries.

Various methods can be applied to quantify the uncertainty based on the output function of graph learning $\mathbf{u}^*$. Let $\mathbf{u}^*(\mathbf{x}_j) = (u_1^*(\mathbf{x}_j), u_2^*(\mathbf{x}_j), \ldots, u_{n_c}^*(\mathbf{x}_j))$ be the output vector of graph learning on $\mathbf{x}_j \in X$. Define the

1. **Smallest Margin [Set09]:** This acquisition function evaluates the uncertainty by the difference of the top-1 and top-2 output values:

$$\mathcal{A}_{\mathrm{SM}}(\mathbf{x}_j) = 1 - \left(u_{k_0}^*(\mathbf{x}_j) - \max_{k=1,2,\ldots,n_c; k\neq k_0} u_k^*(\mathbf{x}_j) \right), \tag{2.40}$$

   where $\mathbf{x}_j \in X_u$, $k_0 = \arg\max_{j=1,2,\ldots,n_c} u_k^*(\mathbf{x}_j)$ is the index of the top-1 predicted class.

2. **Entropy [Sha48]**: This acquisition function measures the uncertainty based on the entropy of the predicted class probabilities:

$$\mathcal{A}_{\mathrm{Entropy}}(\mathbf{x}_j) = -\sum_{k=1}^{n_c} u_k^*(\mathbf{x}_j) \log u_k^*(\mathbf{x}_j). \tag{2.41}$$

3. **Least Confidence [Set09]**: This acquisition function evalates the uncertainty by the difference between 1 and the highest predicted value for a single class:

$$\mathcal{A}_{\mathrm{LC}}(\mathbf{x}_j) = 1 - \max_{k=1,2,\ldots,n_c} u_k^*(\mathbf{x}_j). \tag{2.42}$$

4. **Norm Difference [Set09]**: This acquisition function measures the difference between the predicted soft labels and their one-hot thresholded pseudo labels by calculating the ℓ_2-norm difference:

$$\mathcal{A}_{\mathrm{ND}}(\mathbf{x}_j) = \left\|\hat{\mathbf{u}}_j - \mathbf{u}^*(\mathbf{x}_j)\right\|_2, \tag{2.43}$$

where $\hat{\mathbf{u}}_j$ is the one-hot thresholded vector obtained from $\mathbf{u}^*(\mathbf{x}_j)$.

5. **ℓ_2-Norm [MC23]**: This acquisition function evalates the uncertainty by the ℓ_2-norm of the graph learning output $\mathbf{u}^*$:

$$\mathcal{A}_{\mathrm{L2}}(\mathbf{x}_j) = \|\mathbf{u}^*(\mathbf{x}_j)\|_2 = \sqrt{\sum_{k=1}^{n_c} [u_k^*(\mathbf{x}_j)]^2}. \tag{2.44}$$

This acquisition has the theoretical guarantee of exploring unexplored clusters in the whole graph structure [MC23].

In this thesis, if not specified, the uncertainty acquisition we applied is the **Smallest Margin** uncertainty function, i.e.,

$$\mathcal{A}_{\mathrm{UC}}(\mathbf{x}_j) := \mathcal{A}_{\mathrm{SM}}(\mathbf{x}_j),\ \forall X_j \in X_u. \tag{2.45}$$

2.3.2.2 Variance Optimization (VOpt) Acquisition Function

According to the Bayesian Interpretation of the graph Laplace learning (2.35), the expected prediction error on the unlabeled set X_u can be computed as follows [JH12, MMS22]:

$$\mathbb{E}\left(\sum_{\mathbf{x}_j \in X_u} (\mathbf{u}(\mathbf{x}_j) - \mathbf{u}^*(\mathbf{x}_j))\right) = \mathbf{Tr}\left(L_{uu}^{-1}\right). \tag{2.46}$$

This implies that if we want to select an unlabeled vertex $\mathbf{x}_j \in X_u$ and acquire the label for it, we need to minimize

$$\mathbf{Tr}\left(L_{\hat{u}\hat{u}}^{-1}\right), \tag{2.47}$$

where $L_{\hat{u}\hat{u}}$ is the sub-matrix of L on the rows and columns $X_u \setminus \{\mathbf{x}_j\}$.

Practically, we consider this optimization problem under the MGR model with the parameter γ (Section 2.2.2 rather than the graph Laplace learning. By adding an unlabeled data $\mathbf{x}_j \in X_u$ to the labeled set, the covariance matrix C_{MGR} (2.36) becomes:

$$\begin{aligned} C_{\text{MGR}}^{+\mathbf{x}_j} &= \left(L + \frac{1}{\gamma^2} P^\top P + \frac{1}{\gamma^2} \mathbf{e}_j \mathbf{e}_j^\top\right)^{-1} \\ &\approx V\left(\Lambda + \frac{1}{\gamma^2} V^\top P^\top P V + \frac{1}{\gamma^2} V^\top \mathbf{e}_j \mathbf{e}_j^\top V\right)^{-1} V^\top \\ &= V\left(\Sigma_{\text{MGR}}^{-1} + \frac{1}{\gamma^2} V^\top \mathbf{e}_j \mathbf{e}_j^\top V\right)^{-1} V^\top, \end{aligned} \tag{2.48}$$

where V, Λ are the eigenvectors and eigenvalues (diagonal matrix) of the smallest n eigenvalues of L defined in Section 2.3.1, and Σ is defined by (2.38).

The corresponding variance minimization problem becomes:

$$\min_{\mathbf{x}_j \in X_u} \mathbf{Tr}\left[V\left(\Sigma_{\text{MGR}}^{-1} + \frac{1}{\gamma^2} V^\top \mathbf{e}_j \mathbf{e}_j^\top V\right)^{-1} V^\top\right]. \tag{2.49}$$

According to properties of the trace of a matrix, the orthonormality of the eigenvectors of

L, and the Woodbury matrix identity [Woo50], we have:

$$\mathbf{Tr}\left[V\left(\Sigma_{\mathrm{MGR}}^{-1}+\frac{1}{\gamma^2}V^\top\mathbf{e}_j\mathbf{e}_j^\top V\right)^{-1}V^\top\right]=\mathbf{Tr}\left[\left(\Sigma_{\mathrm{MGR}}^{-1}+\frac{1}{\gamma^2}V^\top\mathbf{e}_j\mathbf{e}_j^\top V\right)^{-1}\right]$$
$$=\mathbf{Tr}(\Sigma_{\mathrm{MGR}})-\frac{1}{\gamma^2+\mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\|\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\|_2^2. \tag{2.50}$$

Since the $\mathbf{Tr}(\Sigma_{\mathrm{MGR}})$ is irrelevant to $\mathbf{x}_j$, we can write the VOpt acuqisition function as:

$$\mathcal{A}_{\mathrm{VOpt}}(\mathbf{x}_j)=\frac{1}{\gamma^2+\mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\|\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\|_2^2,\ \forall\mathbf{x}_j\in X_u, \tag{2.51}$$

which only requires to store the $n\times n$ matrix Σ_{MGR} and the truncated eigenvector matrix V. The computation of $\mathcal{A}_{\mathrm{VOpt}}$ for a single $\mathbf{x}_j\in X_u$ has the time complexity $\mathcal{O}(nN)$.

2.3.2.3 Model-change (MC) Acquisition Function

Model-Change (MC) [MLB20, MB24] is a recently proposed approach that evaluates the potential impact of adding an unlabeled vertex $\mathbf{x}_j\in X_u$ in the graph to the labeled set X_l. The MC acquisition function measures the extent to which the graph-based model, such as would be modified if the point were incorporated with its predicted label. This allows the active learning algorithm to prioritize instances that are likely to have the greatest effect on refining the model's understanding of the underlying data distribution.

To motivate the MC acquisition function, we first modify the standard graph learning model (2.15) into a *look-ahead model* objective:

$$\mathcal{J}^{\mathbf{x}_j,\hat{\mathbf{u}}_j}(U,U^\dagger;\hat{\mathbf{u}}_j)=\frac{1}{2}\langle U,LU\rangle_F+\sum_{\mathbf{x}_i\in X_l}\ell(\mathbf{u}(\mathbf{x}_i),\mathbf{u}^\dagger(\mathbf{x}_i))+\ell(\mathbf{u}(\mathbf{x}_k),\hat{\mathbf{u}}_j), \tag{2.52}$$

where $\mathbf{x}_j\in X_u$ is an unlabeled vertex in the graph, $\hat{\mathbf{u}}_j$ is the pseudo label of $\mathbf{x}_j$, and $\ell=\ell_\gamma$ is the MGR regularization function. Let $\mathbf{u}^*$ be the output of the graph Laplace learning model, then $\hat{\mathbf{u}}_j$ is the one-hot thresholding of the vector $\mathbf{u}^*(\mathbf{x}_j)$. The one-hot thresholding process selects the index of the maximum element in the vector $\mathbf{u}^*(x_j)$ and sets the corresponding element in the resulting one-hot vector $\hat{\mathbf{u}}_j$ to 1, while all other elements are set to 0.

Consider the truncated eigenvalue and eigenvector matrices $\Lambda \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{N\times n}$ defined in Section 2.3.1. Define $A = V^\top U \in \mathbb{R}^{n\times n_c}$ to be the projection of the matrix U onto the eigenvectors of the graph Laplacian. The look-ahead model can be approximated by:

$$\begin{aligned}\mathcal{J}^{\mathbf{x}_j,\hat{\mathbf{u}}_j}(\mathbf{u},\mathbf{u}^\dagger;\hat{\mathbf{u}}_j) &\approx \mathcal{J}_A^{\mathbf{x}_j,\hat{\mathbf{u}}_j}(A,\mathbf{u}^\dagger;\hat{\mathbf{u}}_j)\\ &= \frac{1}{2}\langle A,\Lambda A\rangle_F + \frac{1}{2\gamma^2}\left(\sum_{\mathbf{x}_i\in X_l}\|\mathbf{e}_i^\top VA - \mathbf{u}^\dagger(\mathbf{x}_i)\|_2^2 + \|\mathbf{e}_j^\top VA - \hat{\mathbf{u}}_j\|_2^2\right).\end{aligned} \tag{2.53}$$

Define $A^* = V^T U^*$, and $\hat{A}^{\mathbf{x}_j,\hat{\mathbf{u}}_j}$ as the optimal matrix by solving the problem:

$$\hat{A}^{\mathbf{x}_j,\hat{\mathbf{u}}_j} = \underset{A\in\mathbb{R}^{N\times N_c}}{\arg\min}\, \mathcal{J}_A^{\mathbf{x}_k,\hat{\mathbf{u}}_j}(A,\mathbf{u}^\dagger;\hat{\mathbf{u}}_j). \tag{2.54}$$

The optimal matrix $\hat{A}^{\mathbf{x}_j,\hat{\mathbf{u}}_j}$ can be estimated using a one-step Newton's iteration on the look-ahead objective function (2.53):

$$\hat{A}^{\mathbf{x}_j,\hat{\mathbf{u}}_j} \approx A^* - \frac{1}{\gamma^2 + \mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\left(\mathbf{e}_j^\top VA^* - \hat{\mathbf{u}}_j\right). \tag{2.55}$$

Using the approximation (2.55), the MC acquisition function is given by:

$$\begin{aligned}\mathcal{A}_{\mathrm{MC}}(\mathbf{x}_j) &= \left\|U^{\mathbf{x}_j,\hat{\mathbf{u}}_j} - U^*\right\|_F = \left\|\hat{A}^{\mathbf{x}_j,\hat{\mathbf{u}}_j} - A^*\right\|_F\\ &= \left\|\frac{1}{\gamma^2 + \mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\left(\mathbf{e}_j^\top VA^* - \hat{\mathbf{u}}_j\right)\right\|_F\\ &= \frac{\left\|\mathbf{e}_j^\top VA^* - \hat{\mathbf{u}}_j\right\|_2}{\gamma^2 + \mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\left\|\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\right\|_2.\end{aligned} \tag{2.56}$$

Recall that for the full-rank case, we have $VA^* = U^*$, therefore we can write the MC acquisition function in the form:

$$\mathcal{A}_{\mathrm{MC}}(\mathbf{x}_j) = \frac{\|\mathbf{u}^*(\mathbf{x}_j) - \hat{\mathbf{u}}_j\|_2}{\gamma^2 + \mathbf{e}_j^\top V\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j}\left\|\Sigma_{\mathrm{MGR}}V^\top\mathbf{e}_j\right\|_2. \tag{2.57}$$

where the matrix A^* is eliminated, and we only need to record Σ_{MGR}, V and the optimal solution of the graph Laplace learning U^*. By comparing with the norm difference UC acquisition function (2.43), and the VOpt acquisition function (2.51), the MC acquisition

function (2.57) can be viewed as a modified version of the VOpt acquisition by replacing $\left\|\Sigma_{\mathrm{MGR}} V^{\top} \mathbf{e}_j\right\|_2$ with the UC acquisition function $\mathcal{A}_{\mathrm{ND}}$.

The MCVOpt acquisition function [MMS22] combines the MC (2.57) and VOpt (2.51) acquisition functions according to their similarity:

$$\mathcal{A}_{\mathrm{MCVOpt}}(\mathbf{x}_j) = \frac{\left\|\mathbf{u}^*(\mathbf{x}_j) - \hat{\mathbf{u}}_j\right\|_2}{\gamma^2 + \mathbf{e}_j^{\top} V \Sigma_{\mathrm{MGR}} V^{\top} \mathbf{e}_j} \left\|\Sigma_{\mathrm{MGR}} V^{\top} \mathbf{e}_j\right\|_2^2. \tag{2.58}$$

2.3.3 Query Set Selection

The selection of query set $\mathcal{Q}$ is tricky. In *sequential active learning*, the query set in each iteration is selected as the single most informative instance with the highest acquisition function value, i.e.:

$$Q = \{\mathbf{x}_k\},\ \mathbf{x}_k = \underset{\mathbf{x} \in X_u}{\arg\max}\, \mathcal{A}(\mathbf{x}). \tag{2.59}$$

However, sequential active learning can be inefficient, especially when dealing with large datasets, as it requires updating the model and recomputing the acquisition function after each instance is labeled. Moreover, it does not support parallel labeling by multiple human experts, which could significantly speed up the annotation process.

To address these limitations, *batch active learning* has been proposed. In batch active learning, a query set of batch size $B > 1$ is selected at each iteration, allowing for the parallel labeling of multiple instances and reducing the number of iterations. The goal is to select a batch of informative instances that collectively maximize the expected improvement in the model's performance.

The main challenge in transitioning from sequential active learning to batch active learning is that sequential methods do not account for the inherent redundancy and similarity between unlabeled data points. Applying batch active learning naively often results in the selection of homogeneous batches, where the chosen instances are closely clustered in the embedding space. For instance, selecting the top k instances with the highest acquisition function values may lead to a batch of near-duplicate instances, providing little additional

information to the model. Consequently, the model's learning rate may not significantly improve compared to the sequential case, despite the increased labeling effort at each iteration. To address this issue, batch active learning methods must explicitly encourage diversity in the selected batch while maintaining high acquisition function values.

Another critical aspect of batch active learning is the computational complexity of optimizing the query set $\mathcal{Q}$ with size $|\mathcal{Q}| = B$. Finding the optimal subset of unlabeled instances is a combinatorial problem with a complexity of $O(N^B)$, where N is the total number of unlabeled instances. This high computational cost severely limits the feasible batch sizes, making it impractical for real-world applications. To mitigate this problem, heuristic approaches should be employed to efficiently approximate the optimal query set, striking a balance between the acquisition function value and the diversity of the selected instances.

One typical batch active learning approach utilizes greedy algorithms, especially the lazy greedy algorithm, in combination with submodular optimization techniques to efficiently select informative and diverse batches [GB10, JH12, CK13, WIB15, SS18]. These greedy methods iteratively select a query set $\mathcal{Q}$ by mimicking sequential active learning but without the need to obtain ground-truth labels from an oracle at each step.

Define the initial acquisition function by $\mathcal{A}_1 = \mathcal{A}$ and the initial query set $\mathcal{Q}_0 = \emptyset$. At step $K \leq B$ of the greedy algorithm, it updates the query set $\mathcal{Q}_k$:

$$\mathcal{Q}_k = \mathcal{Q}_{k-1} \cup \{\mathbf{x}^*\}, \quad \mathbf{x}^* = \underset{\mathbf{x} \in X_u \setminus \mathcal{Q}_{k=1}}{\arg\max}\ \mathcal{A}(\mathbf{x}). \tag{2.60}$$

Then it augments the labeled set X_l with the current query set $\mathcal{Q}_k$ with its corresponding pseudo label, obtained by the graph learning classifier. Using the augmented labeled set $X_l^k = X_l \cup \mathcal{Q}_k$, we update the predictions on the unlabeled set $X_u \setminus \mathcal{Q}_k$ and calculate the acquisition function $\mathcal{A}_k$ on $X_u \setminus \mathcal{Q}_k$. This process is repeated until step B when the final query set $\mathcal{Q} = \mathcal{Q}_B$ has the size B. Such a greedy algorithm is guaranteed to be near-optimal to the submodular set function, which can be considered as a set-version acquisition function [NWF78].

Another type of the batch active learning method addresses the computational complexity and potential redundancy issues by restricting the evaluation of the acquisition function $\mathcal{A}$ to a smaller candidate set $\hat{X}^U \subset X_u$, where $\hat{X}^U$ is chosen uniformly at random from the unlabeled pool X_u [WY13, GIG17, AZK20]. Instead of considering all unlabeled instances, these methods select the batch $\mathcal{Q} \subset X_u$ as the top maximizers of the acquisition function within this reduced set. This approach offers two significant advantages. First, by limiting the evaluation of $\mathcal{A}$ to a subset $\hat{X}^U$, where $|\hat{X}^U| \ll |X_u|$, the computational cost is greatly reduced, making the batch selection process more efficient. Second, the random selection of $\hat{X}^U$ helps to mitigate the issue of "redundant" calculations, as the maximizers of $\mathcal{A}$ over $\hat{X}^U$ are less likely to be clustered together, promoting diversity within the selected batch.

As one of the main contributions of this thesis, we propose novel graph-based core-set and batch active learning methods, which will be introduced in detail in Chapter 3. Our method is highly efficient and achieves performance comparable to sequential active learning. Before our work, several graph-based batch active learning methods have been proposed. These methods typically leverage the graph structure to capture the similarity and diversity among instances, guiding the selection of informative and representative batches [ZLG03, JH12, CZC17].

CHAPTER 3

Novel Batch Active Learning Approaches with Application to SAR and Hyperspectral Imagery

This chapter reuses materials from the author's publications [CCT23, CMB23a, BCH23]. Content from [CCT23] is reproduced with permission from SPIE, while content from [CMB23a] is used under the Creative Commons CC BY license. IEEE copyrighted material from [BCH23] is reused in this chapter, with the approval of the senior author Andrea L. Bertozzi and following the requirements outlined by IEEE for thesis/dissertation reuse. [1]

Active learning is a powerful technique that improves the performance of machine learning methods by judiciously selecting a limited number of unlabeled data points to query for labels, to maximally improve the underlying classifier's performance [Set09, Das11, MMS22, MB24]. As mentioned in Section 2.3.3, sequential active learning methods, which select a single data point to query in each iteration, can be inefficient and computationally expensive, especially for large datasets. Batch active learning methods, on the other hand, select multiple data points to query in each iteration, potentially improving efficiency. However, batch query selection poses several challenges, such as ensuring diversity among the selected data points, avoiding redundancy, and maintaining the informativeness of the queried labels. Addressing these challenges is crucial for the effectiveness of batch active learning methods.

Our work in this chapter makes the following innovations and contributions:

1. We propose novel methods for batch active learning, including Dijkstra's Annulus

[1] ©2023 IEEE. Reprinted, with permission, from [BCH23]

Core-Set (DAC) for core-set generation and LocalMax for batch sampling, which effectively overcome challenges in existing approaches and improve both the accuracy and efficiency of the learning process;

2. We construct comprehensive pipelines that integrate transfer learning-based feature embedding, graph learning, and batch active learning techniques to achieve high-quality and efficient classification of SAR images and segmentation of hyperspectral images with limited labeled data;

3. We demonstrate the versatility and effectiveness of our approach across different domains, with remarkable results achieved using limited labeled data in both SAR image classification and hyperspectral image segmentation tasks.

We apply our proposed DAC and LocalMax methods to two distinct domains: synthetic aperture radar (SAR) image classification and multi- or hyperspectral image segmentation. In the context of SAR data classification, we develop a pipeline based on transfer learning feature embedding, graph learning, DAC, and LocalMax to classify the FUSAR-Ship and OpenSARShip datasets. Our approach outperforms state-of-the-art CNN-based methods while achieving nearly identical accuracy as sequential active learning, but with improved efficiency proportional to the batch size [CCT23].

For pixel/patch neighborhood multi- or hyperspectral image segmentation, we provide a graph-based batch active learning pipeline that incorporates our DAC and LocalMax methods. The pipeline selects a collection of unlabeled pixels that satisfy a graph local maximum constraint for the active learning acquisition function, which determines the relative importance of each pixel to the classification. Graph learning, when used as a semi-supervised learning (SSL) method, performs well for classification tasks with a low label rate. Our approach not only improves accuracy but also greatly reduces the number of labeled pixels needed to achieve the same level of accuracy compared to randomly selected labeled pixels [CMB23a].

The successful application of our proposed DAC and LocalMax methods to both SAR image classification and multi- or hyperspectral image segmentation demonstrates their versatility and effectiveness in different domains. By leveraging these novel batch active learning techniques, along with patch-neighborhood analysis and graph-based learning, we significantly accelerate the active learning process while maintaining high accuracy. Our methods reduce the number of iterations required to achieve a desired performance level, leading to a substantial decrease in the overall labeling effort and computational cost. This improved efficiency is particularly valuable when dealing with large-scale datasets or time-sensitive applications. Moreover, our approach not only speeds up the learning process but also surpasses the performance of traditional methods, showcasing the effectiveness of our contributions in advancing the field of active learning for image analysis.

The codes about methods proposed in this chapter are available on GitHub [2].

3.1 Background

In the previous Chapter 2, we reviewed the graph learning and active learning approaches. In this section, we introduce the background of the SAR image and hyperspectral image with the related application of graph learning and active learning on these two kinds of datasets.

3.1.1 Classification on Synthetic Aperture Radar (SAR) Imagery

Synthetic Aperture Radar (SAR) is a powerful remote sensing technology that plays a vital role in Automatic Target Recognition (ATR) [AMZ18, LSS19, IID21, MMS22]. SAR systems utilize a moving platform, such as an aircraft or satellite, to transmit and receive radio signals repeatedly, effectively simulating a large radar dish. This process enables the creation of high-resolution images, making SAR a valuable tool for various applications. The SAR community

[2]Source Code: `https://github.com/chapman20j/SAR_BAL`

has established several benchmark datasets to facilitate research and development in this field. For instance, the MSTAR dataset contains SAR images of land-based vehicles [AD], while the OpenSARShip and FUSAR-Ship datasets focus on SAR images of different types of ships at sea [HLL17, HAS20]. Three samples from these three datasets are shown in Figure 3.1. These datasets serve as standard references for evaluating and comparing the performance of object recognition algorithms.

Semi-supervised learning (SSL) has emerged as a promising approach for SAR classification due to its recent success and high data efficiency [ZGL03, BLR04, BF12, BM19, MMS22]. Unlike traditional supervised learning methods that rely solely on labeled data, SSL algorithms leverage the geometric structure of the entire dataset, including both labeled and unlabeled samples. This capability makes SSL particularly valuable in scenarios where labeled data is scarce, as is often the case in SAR image classification. In this section, we employ graph-based Laplace learning as the underlying classifier, which exploits the inherent geometry of the SAR data to improve classification performance.

To effectively harness the geometric structure of SAR images, it is crucial to extract meaningful features that capture the essential characteristics of the data. Convolutional Neural Networks (CNNs) [LBD89] have demonstrated remarkable success in image classification tasks, owing to their ability to learn rich, hierarchical representations of visual information. Moreover, CNNs are well-suited to handle the significant noise present in SAR images, making them an attractive choice for feature extraction in this domain. Previous research [MMS22] has successfully utilized Convolutional Variational Auto-Encoders (CNNVAEs) [KW13, PGH16] to embed SAR data into a lower-dimensional space, facilitating more efficient and effective classification.

In addition to CNNVAEs, transfer learning has emerged as another promising approach for obtaining useful image features in SAR classification [AMZ18, BKS21, IID21]. Transfer learning involves training a neural network on a similar dataset for which abundant labeled data is available and then adapting the learned features to the target task. By leveraging

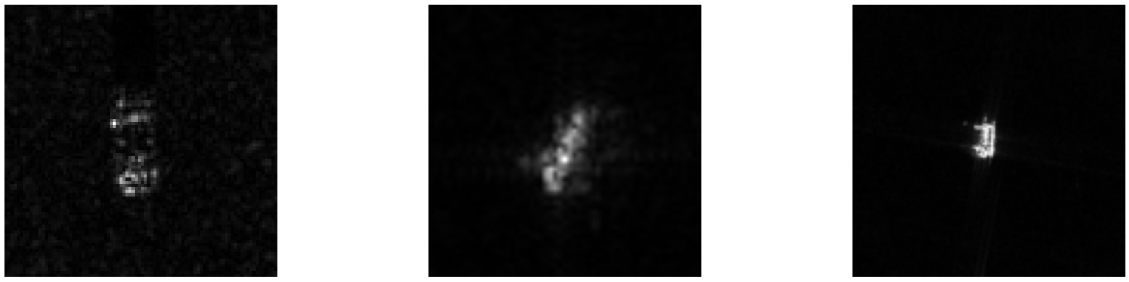


Figure 3.1: Three samples of SAR images. From left to right, the images show vehicles or ships from MSTAR [AD], OpenSARShip [HLL17], and FUSAR-Ship [HAS20].

the knowledge gained from the source domain, transfer learning can significantly reduce the amount of labeled data required for training on the target SAR dataset. The later convolutional layers of the pre-trained network contain valuable image features that can be directly utilized for SAR classification. Alternatively, these features can be fine-tuned by replacing the fully connected layers with linear layers and further training on the current dataset to obtain more task-specific representations. In this paper, we explore three feature extraction techniques: CNNVAE, transfer learning without fine-tuning, and transfer learning with fine-tuning, to enhance the performance of our SSL-based SAR classification framework.

Recent advances in ATR can be broadly categorized into two main approaches: supervised learning and SSL. Deep learning has been at the forefront of these advancements, enabling the extraction of valuable image features. Zhang et al. developed a supervised deep learning model called Hog-ShipCLSNet for classifying the OpenSAR Ship and FUSAR-Ship datasets [ZZK21]. Other notable supervised learning methods employ transfer learning [PY09] from simulated SAR datasets [AMZ18, IID21]. While simulating SAR data can alleviate the labeling burden on experts, it poses significant challenges due to the distribution shift between the simulated and real-world datasets [LSS19]. In the realm of SSL, Miller et al. made a significant contribution by applying a CNNVAE and graph-based sequential active learning to classify images in the MSTAR dataset [MMS22], achieving state-of-the-art performance.

3.1.2 Segmentation on Multi- or Hyperspectral Imagery

Image segmentation is a fundamental problem in the field of machine learning and computer vision, with applications spanning across various domains, including remote sensing [KSH12, LBH15]. In remote sensing, image segmentation plays a crucial role in extracting meaningful information from satellite and aerial imagery [CH16, ZZD16], enabling a wide range of applications such as land cover classification [KLS17], change detection [Zhu17], and environmental monitoring [PWS16].

Multispectral imaging (HSI) and hyperspectral imaging (HSI) technologies have revolutionized the field of remote sensing by providing rich spectral information across hundreds or even thousands of narrow spectral bands [BPC13]. This high-dimensional data allows for the discrimination of different materials and objects based on their unique spectral signatures, which is not possible with traditional RGB imagery [CTB13]. Image segmentation techniques are particularly valuable in the context of multi- and hyperspectral remote sensing, as they allow for the automatic delineation of distinct regions or objects within the image based on their spectral characteristics [LSF19]. Accurate segmentation of MSI and HSI is essential for a wide range of applications, including precision agriculture [Mul13], mineral exploration [MWV12], and environmental monitoring [XSY08].

Several approaches have been proposed for image segmentation in remote sensing, ranging from traditional methods to more recent deep learning-based techniques. One older approach involves partial differential equation (PDE)-based methods, which segment an image by solving a PDE on the image numerically, based on minimization of an energy functional [MS89, KWT88, CV01, BEV07]. More recently, graph-based methods have also been developed for both semi-supervised and unsupervised learning on image processing [GO09, MSB14, MKB13, BPB20, BBT18, HSB15, HLP13, BF12, GMB14, MMK17, CBC14]. Another common choice is the neural network methods, including CNN [LBD89] and GCN [WSZ19, TNX21], with trainable convolutional filters optimized by minimizing the difference

between predicted and ground-truth labels.

3.2 Core-Set Selection and Batch Active Learning

As introduced in Section 2.3, active learning is an iterative process that selects a query set $\mathcal{Q}$ to obtain labels in each iteration. As discussed in the query set selection (Section 2.3.3), sequential active learning chooses a single unlabeled vertex in the graph for each query (the case $|\mathcal{Q}| = 1$), while batch active learning selects multiple unlabeled vertices as the query set (the case $|\mathcal{Q}| > 1$). In most cases, the human labeling time represents the most significant bottleneck in the process. Sequential active learning limits the human labeling process by allowing only one point to be labeled at a time. It is beneficial for the active learning procedure to return a batch of points, enabling multiple humans/teams to work in parallel to label the data. For instance, consider the task of labeling 200 points with a team of 10 people. In the sequential case $|\mathcal{Q}| = 1$, labeling necessitates 200 human queries. However, in the batch case with $|\mathcal{Q}| = 10$, only 20 human queries are needed, and human experts can label the data simultaneously. This parallel labeling approach reduces the total labeling time by an order of magnitude compared to the sequential case.

Many methods rely on using the current predictions of the model to evaluate the uncertainty, variance, or other criteria to quantify the information gained by labeling a new data point. These methods are based on a critical assumption: the model possesses sufficient information to determine which data would be most beneficial for its learning process. This leads to two important requirements: constructing a good set of initial labels, called a *core-set*, for the early stages of active learning, and quantifying information shared by unlabeled points in a batch. It is crucial to construct a good core-set to enable the model to make accurate estimates of the acquisition function early in training when model accuracy is relatively low [MKS13, SS18]. The core-set construction can have a long-lasting impact on the performance of the active learning procedure, and it is essential that the method

adequately explores the data before exploiting knowledge with active learning. Fortunately, graph-based active learning methods perform well with relatively small amounts of labeled data [MMS22].

The main challenge in transitioning from sequential to batch active learning is that naive batch selection often results in redundant points, leading to inefficient learning. As discussed in Section 2.3.3, heuristics should be employed to efficiently select diverse, informative batches while considering the computational complexity of optimizing the query set $\mathcal{Q}$.

3.2.1 Dijkstras Annulus Core-Set (DAC) Selection

The primary objective of core-set selection is to extensively explore the data, enabling the active learning process to achieve optimal performance. We introduce Algorithm 2 for core-set selection, as it produces a core-set that is approximately evenly distributed throughout the dataset. Given a set of feature vectors X, we construct a graph $G = (X, W)$ in accordance with Section 2.2.1. The algorithm iteratively chooses nodes in X to form a core-set, ensuring that all points are separated by a distance of at least r but no more than R from any other point. At each iteration of the core-set selection process, assuming X_l is the currently chosen vertex set (i.e., the current labeled set), the algorithm generates an *annular set* $X_C \subset X$ and a *seen set* $X_S \subset X$:

$$X_C = \left[\bigcup_{\mathbf{x}\in X_l} B_R(\mathbf{x})\right] \setminus \left[\bigcup_{\mathbf{x}\in X_l} B_r(\mathbf{x})\right], \quad X_S = \bigcup_{\mathbf{x}\in X_l} B_r(\mathbf{x}), \tag{3.1}$$

where

$$B_r(\mathbf{x}) = \{\mathbf{y} \in X : \ d_G(\mathbf{x}, \mathbf{y}) < r\} \tag{3.2}$$

and $d_G(\mathbf{x}, \mathbf{y})$ represents the distance between $\mathbf{x}$ and $\mathbf{y}$, calculated using Dijkstra's algorithm [Dij59]. The annular set contains the points from which the algorithm may choose at each stage, while the seen set consists of points that the algorithm can no longer select.

Algorithm 2 Dijkstras Annulus Core-Set (DAC)

Require: Graph $G = (X, W)$, initial labeled set X_l, inner radius r and outer radius R

Compute annular set from current labeled set

Initialize: Annular set $X_C = \emptyset$ and seen set $X_S = \emptyset$.

Ensure: The core-set X_l.

for $\mathbf{x} \in X_l$ **do**

  Compute $B_r(\mathbf{x}), B_R(\mathbf{x})$

  $X_S \leftarrow X_S \cup B_r(\mathbf{x})$

  $X_C \leftarrow (X_C \cup B_R(\mathbf{x})) \setminus B_r(\mathbf{x})$

end for

Iterative process updating the core-set X_l

while $X_S \neq X$ **do**

  if $X_C = \emptyset$ **then**

    pick $\mathbf{x} \in X \setminus X_S$ uniformly at random

  else if $X_C \neq \emptyset$ **then**

    pick $\mathbf{x} \in X_C$ uniformly at random

  end if

  Compute $B_r(\mathbf{x})$, $B_R(\mathbf{x})$

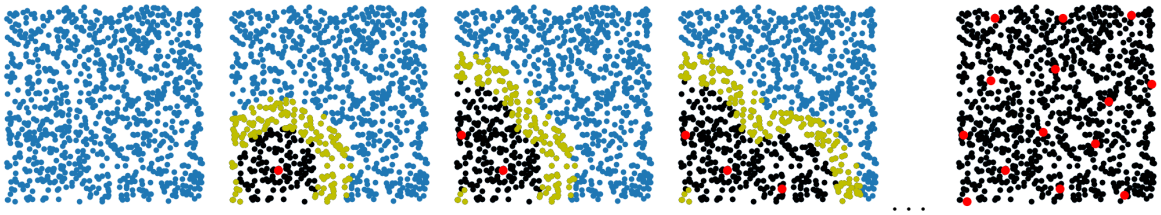
  $X_l \leftarrow X_l \cup \{\mathbf{x}\}$

  $X_S \leftarrow X_S \cup B_r(\mathbf{x})$

  $X_C \leftarrow (X_C \cup B_R(\mathbf{x})) \setminus B_r(\mathbf{x})$

end while

Algorithm 2 proceeds by randomly selecting $\mathbf{x} \in X_l$ and updating X_l, X_S, X_C. Iterating this process yields a relatively uniform coverage of the data. It is important to note that it may not always be feasible to select $\mathbf{x} \in X_C$ before $X_S = X$. In such cases, the algorithm randomly jumps to another data point outside the seen set. This situation can arise if R is too small or if the data contains well-separated clusters. The output of Algorithm 2 is



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4 (e) Iteration 14

Figure 3.2: An example of the sampling process of the DAC algorithm with an outer density radius of 0.3. The dataset is generated by sampling uniformly at random in the unit square. The blue, black and gold points denote the unseen points, the seen points and the points in the annular set, respectively. In iteration 0, the annular set is empty and the unseen set isn't empty. This means the algorithm picks a point at random from the unseen points to add to the core-set. In subsequent iterations, the algorithm picks a point at random from the annular set. It then updates the annular region as described in Algorithm 2. This process terminates at iteration 14 when the entire dataset is the seen set. The set of red points in panel (e) is the output DAC core-set, which is nearly uniformly distributed in the whole dataset.

the core-set X_l, which is utilized as the initial labeled set in the active learning process. Figure 3.2 illustrates an example of the algorithm applied to a simple dataset.

Theorem 8. *Let X_l be the core-set output from our DAC Algorithm 2 with the radius parameters r, R. Then we have:*

1. *For any $\mathbf{x} \in X$, there exists $\mathbf{x}_0 \in X_l$ such that $d_G(\mathbf{x}, \mathbf{x}_0) < r$.*
2. *If the initial core-set X_l^0 satisfies that for any $\mathbf{x}_1 \neq \mathbf{x}_2 \in X_l^0$, $d_G(\mathbf{x}_1, \mathbf{x}_2) \geq r$. Then for any $\mathbf{x}_1 \neq \mathbf{x}_2 \in X_l$, $d_G(\mathbf{x}_1, \mathbf{x}_2) \geq r$*

Proof. 1. When the DAC Algorithm 2 terminates, we have $X_S = X$, i.e.,

$$\bigcup_{\mathbf{x} \in X_l} B_r(\mathbf{x}) = X. \tag{3.3}$$

For any $\mathbf{x} \in X$, there exists $\mathbf{x}_0 \in X_l$ such that $\mathbf{x} \in B_r(\mathbf{x}_0)$, which is equivalent to $d_G(\mathbf{x}, \mathbf{x}_0) < r$.

2. For each step of the DAC Algorithm 2, we choose a new

$$\mathbf{x} \in X_C = \left[\cup_{\mathbf{x} \in X_l} B_R(\mathbf{x})\right] \setminus \left[\cup_{\mathbf{x} \in X_l} B_r(\mathbf{x})\right], \tag{3.4}$$

to update the current core-set X_l. This implies that if we write the output $X_l = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_{|}X_l|}\}$, we have

$$d_G(\mathbf{x}_{i_j}, \mathbf{x}_{i_k}) \geq r, \quad \forall j = 1, 2, \ldots, k-1, \ k = 1, 2, \ldots, |X_l|. \tag{3.5}$$

This is equivalent to

$$d_G(\mathbf{x}_1, \mathbf{x}_2) \geq r, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in X_l, \ \mathbf{x}_1 \neq \mathbf{x}_2. \tag{3.6}$$

□

Remark 9. *Here are some improvements to the radius parameter selection for the Algorithm 2:*

1. *To further streamline the parameter selection process, practically, we set $r = R/2$, effectively reducing the number of parameters to be tuned.*

2. *It is worth noting that this algorithm can also be employed with adaptively determined values of r based on the density of the data surrounding a point, called the density radius. For instance, in each step of the annulus core-set algorithm 2, we update the radius r such that around 5% of the data points fall within $B_r(\mathbf{x}_{last})$, where $\mathbf{x}_{last}$ is the latest vertex selected.*

Utilizing the density radius leads to increased exploration in high-density regions and reduced exploration in low-density areas. This approach enables the core-set to prioritize the regions where the majority of the data is concentrated, effectively capturing the underlying

data distribution. By focusing on high-density areas, the algorithm ensures that the core-set is representative of the most informative and relevant portions of the dataset. This is particularly beneficial when dealing with imbalanced or non-uniformly distributed data, as it prevents the core-set from being dominated by outliers or sparse regions. Moreover, the density-based covering is independent of the average distances between data points, which minimizes the need for extensive parameter tuning. This adaptability makes the algorithm more robust and applicable to a wide range of datasets without requiring manual adjustments to the radius parameters.

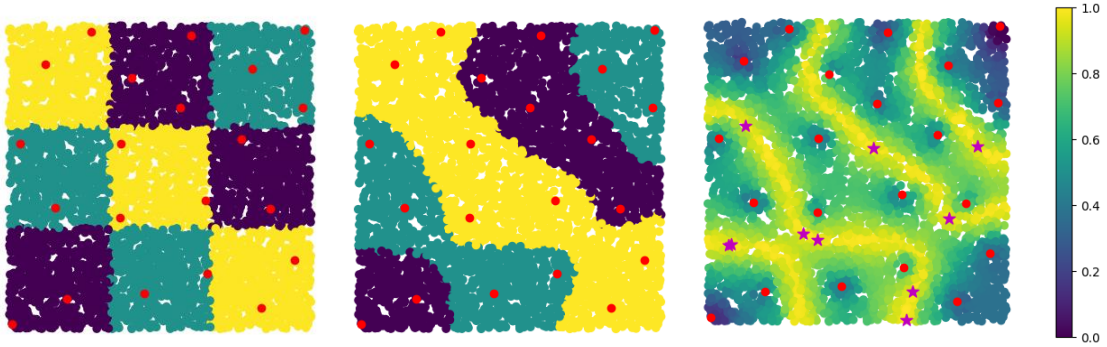
3.2.2 LocalMax Batch Active Learning

We propose a novel batch active learning approach, named LocalMax. Based on a feature vector set X and the corresponding similarity graph $G = (X, W)$, LocalMax selects a query set of multiple nodes that satisfy the local maximum condition (Definition 10) on the graph G from the candidate set. Informally, a node is a local maximum of a function on the nodes if and only if its function value is at least that of its neighbors.

Definition 10 (**Local Max of a Graph Node Function**). *Consider a KNN similarity graph $G = (X, W)$, where X is the set of nodes and W is the edge weight matrix. For a graph node function $\mathbf{u} : X \to \mathbb{R}$, $\mathbf{x}_i \in X$ is a **local maximum node** if and only if for any $\mathbf{x}_j$ adjacent to $\mathbf{x}_i$, $\mathcal{A}(\mathbf{x}_i) \geq \mathbf{u}(\mathbf{x}_j)$. Equivalently, $\mathbf{x}_i \in X$ is a local maximum if and only if:*

$$\mathbf{u}(\mathbf{x}_i) \geq \mathbf{u}(\mathbf{x}_j), \ \forall j \ s.t. \ W_{ij} > 0. \tag{3.7}$$

Assuming a batch size B, at iteration k of the active learning process (Algorithm 1), LocalMax selects the query set $\mathcal{Q}_k$ as the top-B local maximums in the candidate set (unlabeled set) X_u (as detailed in Algorithm 3). Note that we need to extend the acquisition



(a) Ground Truth (b) Classifier (c) Acquisition Values

Figure 3.3: An example of DAC and LocalMax on the checkerboard dataset. In all panels, red points denote the labeled core-set generated by DAC. Panel (a) shows the ground truth classification. Panel (b) shows the classification results of Laplace learning based on the labeled core-set. Panel (c) shows the heatmap of the uncertainty acquisition function evaluated on the dataset. For the uncertainty acquisition function, high acquisition values concentrate near the decision boundary. In panel (c), the purple stars denote points in the query set returned by LocalMax with a batch size of 10.

function $\mathcal{A}_k$ defined on X_u to a graph node function $\hat{\mathcal{A}}_k$ by:

$$\hat{\mathcal{A}}_k = \begin{cases} \mathcal{A}_k(\mathbf{x}), \ \mathbf{x} \in X_u, \\ 0, \ \mathbf{x} \in X_l. \end{cases} \tag{3.8}$$

LocalMax benefits from many useful properties including simplicity, efficiency, and its grounding in well-studied sequential acquisition functions. Building this acquisition function on sequential active learning allows us to borrow properties from the sequential acquisition functions. Controlling for local maxes enforces a minimum pairwise distance between points in the query set, which counteracts the redundancy seen in naively optimizing sequential acquisition functions.

LocalMax also maintains good computational complexity. The computational complexity of Algorithm 3 is $\mathcal{O}(KN)$ where N is the number of nodes in the graph and K is the KNN

Algorithm 3 LocalMax Batch Active Learning

This is the query set selection in one iteration of the active learning process (Algorithm 1).

Require: A KNN graph $G = (X, W)$. The current labeled set X_l and candidate set (unlabeled set) $X_u = X \setminus X_l$. The current acquisition function $\mathcal{A} : X_u \to \mathbb{R}^+$. Batch size B.

Ensure: The query set $\mathcal{Q}$.

Initialize: Extend the domain of $\mathcal{A}$ from X_u to X by defining $\mathcal{A}(\mathbf{x}_j) = 0, \forall j \in X_l$. $\mathcal{Q} = \emptyset$. $S = X_u$

while $S \neq \emptyset$ and $|\mathcal{Q}| < B$ **do**

  $\mathbf{x}_k \leftarrow \arg\max_{\mathbf{x} \in S} \ \mathcal{A}(\mathbf{x})$

  $N(\mathbf{x}_k) = \{\mathbf{x}_j \in X : W_{jk} > 0\}$

  if $\mathcal{A}(\mathbf{x}_k) \geq \mathcal{A}(\mathbf{x}_j), \forall j \in N(\mathbf{x}_k)$ **then**

    $Q \leftarrow Q \cup \{\mathbf{x}_k\}$

  end if

  $S \leftarrow S \setminus N(\mathbf{x}_k)$

end while

parameter used when constructing the graph (Section 2.2.1). In practice, K is much smaller than N, so the computational complexity is $O(N)$. Let $\mathcal{G}(N)$ denote the model fitting time for Laplace learning on a graph G with N nodes and let $\mathcal{H}$ denote the human labeling time for each node. In the case that the weight matrix W is sparse and the graph Laplacian matrix L is well-conditioned, we have $\mathcal{G}(N) = \mathcal{O}(N)$. The human labeling time, $\mathcal{H}$, is typically much greater than $\mathcal{O}(N)$ since labeling SAR data can take significantly more time than is required by the rest of the active learning pipeline. Since human labeling can be processed in parallel, fewer batches are required to label the same amount of data. Consider the active learning process that samples in total M nodes to obtain ground-truth labels. With $\mathcal{G}(N) = O(N)$ and $\mathcal{H} \gg O(N)$, the time complexities of sequential active learning and LocalMax batch

active learning with batch size B are:

$$\text{Sequential Active Learning:} \quad M \times \mathcal{O}(\mathcal{G}(N) + \mathcal{H}) = M\mathcal{H}, \tag{3.9}$$

$$\text{LocalMax Batch Active Learning:} \quad M/B \times \mathcal{O}(\mathcal{G}(N) + \mathcal{O}(N) + \mathcal{H}) = \frac{M}{B}\mathcal{H}. \tag{3.10}$$

LocalMax provides a B times speed up based on sequential active learning which is observed both theoretically and in practice as shown in the following experiment section in this chapter.

3.3 Feature Extraction and Preprocessing

In this section, we introduce two different approaches for extracting features from Synthetic Aperture Radar (SAR) images, Multispectral Images (MSI), and Hyperspectral Images (HSI). These methods are tailored to the specific characteristics and tasks associated with each image type.

For SAR images, which are typically used for classification tasks, the raw features are individual images. To process these images into lower-dimensional features, we consider utilizing Convolutional Neural Network Variational Autoencoders (CNNVAE) or transfer learning techniques. These methods allow us to effectively capture the essential information from the SAR images while reducing the dimensionality of the feature space.

On the other hand, MSI and HSI are commonly used for image segmentation tasks, where the feature vectors are associated with individual pixels. In this context, we employ the concept of non-local means neighborhood patches as feature vectors. By considering the local neighborhood around each pixel, we can capture the spatial and spectral information present in the MSI and HSI data. This approach enables us to represent each pixel with a feature vector that encodes its local context and facilitates accurate segmentation.

3.3.1 Neural Network Feature Embedding for SAR Imagery

The effectiveness of our semi-supervised learning methods relies on a meaningful representation of data, where distances between data points reflect their similarity. CNN architectures have proven to be highly effective in processing image data. In this work, we leverage CNNVAEs from previous research [MMS22] and employ transfer learning techniques using pre-trained PyTorch CNN models to process the SAR datasets. We designate a convolutional layer near the end of the neural network as the *feature layer*, as it captures intricate features of the dataset. The outputs of the feature layer are referred to as *feature vectors*, which are subsequently utilized in our graph construction process, enabling graph-based learning. Figure 3.4 illustrates the transfer learning approach, with the orange layers representing the feature layers.

Transfer learning involves repurposing a neural network that has been trained on a similar dataset and task for a new task. One approach is to directly utilize the parameters from a pre-trained neural network without further modification. Alternatively, fine-tuning the parameters of the original neural network with respect to the new dataset can be employed. We refer to these methods as *zero-shot* transfer learning and *fine-tuned* transfer learning, respectively. Figure 3.4 provides a detailed illustration of the fine-tuned transfer learning process. In contrast, CNNVAEs operate by first encoding the dataset and then decoding it. The neural network architecture is designed to compress the data into a lower-dimensional space during the encoding phase. This compression forces the neural network to learn meaningful image features that enable accurate reconstruction of the original image. In the CNNVAE approach, the feature layer is selected as the final CNN layer in the encoder portion of the network.

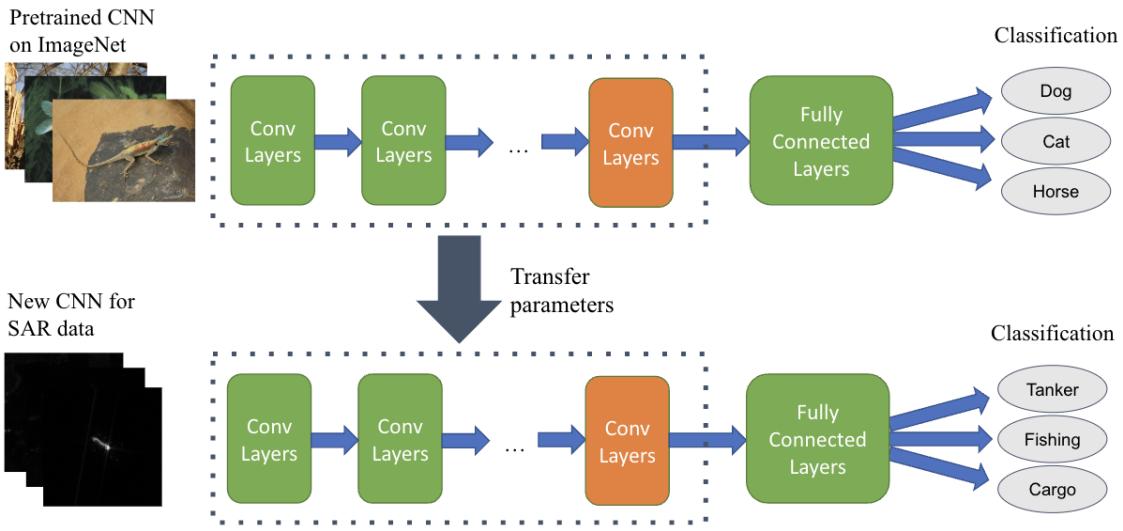


Figure 3.4: Flowchart for fine-tuned transfer learning. The parameters in the convolutional layers (contained in dotted boxes) of the pretrained CNN are transferred to the new CNN. In fine-tuned transfer learning, all the transferred parameters are trained for a few iterations on the new dataset. The training occurs by first adding new fully connected layers at the end of the neural network and performing supervised learning with the new dataset. When training is complete, only the layers in the dotted boxes are kept for the embedding process. The orange layer denotes the feature layer, and the outputs of this layer provide the feature vectors used later in the pipeline.

3.3.2 Non-local Means Feature Extraction for MSI and HSI

For the image segmentation task, the first step is to associate each pixel with a feature vector that captures its local neighborhood information. While using the pixel values of all channels as the feature vector is straightforward, incorporating neighborhood information can enhance the feature representation.

For a pixel indexed by i, we consider a $(2k+1) \times (2k+1)$ neighborhood patch P_i centered at the pixel. If the pixel is near the image boundary, reflection padding is applied to expand the image before extracting the neighborhood patch. Inspired by the non-local

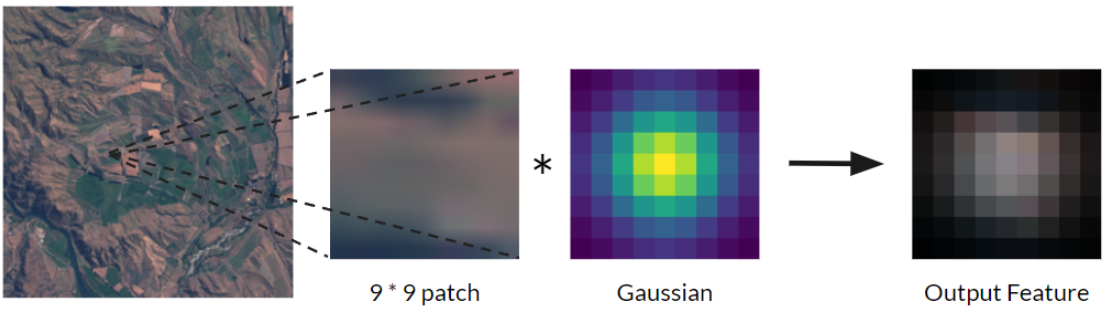


Figure 3.5: The feature extraction process for a single pixel. The feature vector is a Gaussian-weighted patch centered on the pixel.

means method [BCM05], we apply a $(2k+1) \times (2k+1)$ discrete Gaussian kernel $\mathcal{K}_G$ with $\sigma = k/2$ to the neighborhood patch. The Gaussian kernel is defined as:

$$\mathcal{K}_G(i,j) = \frac{\alpha}{2\pi\sigma^2} \exp\left(-\frac{(i-k-1)^2 + (j-k-1)^2}{2\sigma^2}\right), \tag{3.11}$$

where α is a constant such that $\sum_{i,j=1}^{2k+1} \mathcal{K}_G(i,j) = 1$. The weighted patch is then computed as:

$$P_i^w(i,j,l) = P_i(i,j,l)\mathcal{K}_G(i,j),$$

for each pair of pixels $i, j = 1, 2, \ldots, 2k+1$, and $l = 1, 2, \ldots, C$. This feature extraction process is illustrated in Figure 3.5. The non-local means weighting process is applied to each of the C channels in the image. The weighted patches are flattened and concatenated to form the non-local means feature vector for pixel i. The dimension of the resulting feature vector is $d = C(2k+1)^2$.

By extracting the non-local means feature vector for each pixel, we obtain the feature vector set X ($|X| = N$), where N is the total number of pixels in the image. This feature vector set is then used to construct a similarity graph $G = (X, W)$ based on the KNN method, as described in Section 2.2.1. The sparse similarity weight matrix W captures the connectivity between pixels based on their feature vectors.

On the graph G, we randomly initialize a labeled set and apply the LocalMax batch active learning to select a labeled set X_l according to Section 2.3 and 3.2.2. Finally, we predict the

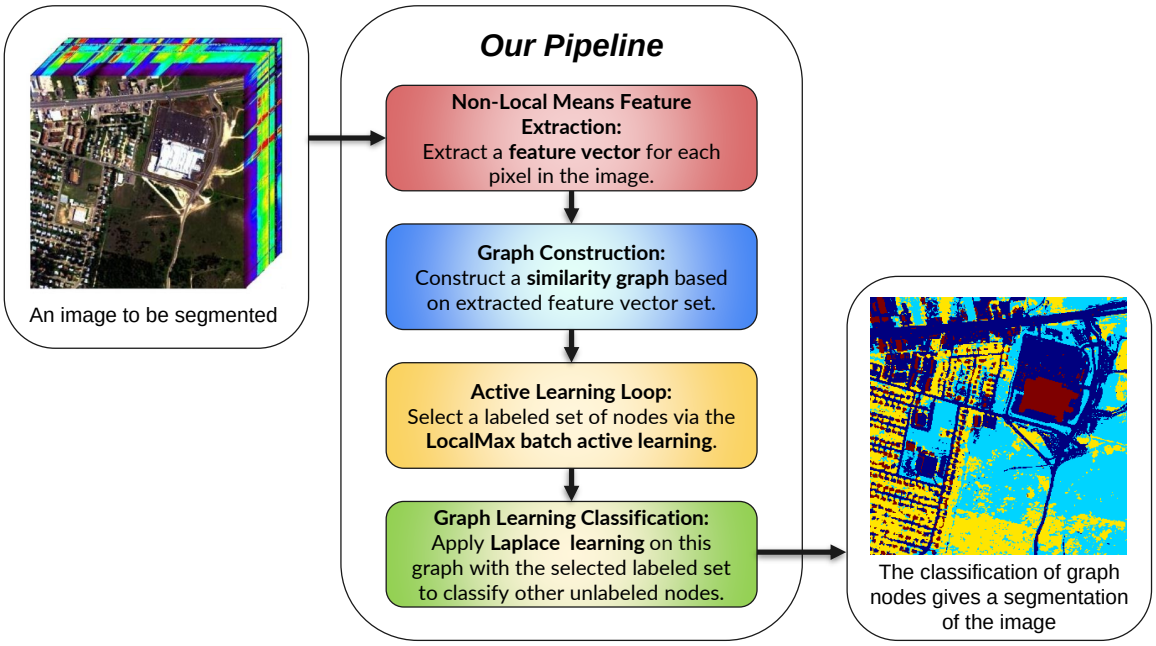


Figure 3.6: Our graph-based active learning pipeline for the image segmentation task. Red box: feature extraction; Blue box: Graph Construction (Section 2.2.1); Yellow box: Batch Active Learning (Section 2.3 and 3.2.2); Green box: Graph Learning (Section 2.2.2).

labels of unlabeled nodes $X_u = X \setminus X_l$ in graph G with the graph Laplace learning classifier based on the selected labeled node set X_l. This node classification on G gives a segmentation on the given image. The flowchart of our pipeline is Figure 3.6.

3.4 Experiments and Results: SAR Image Classification

We now present the results of our active learning experiments on the MSTAR [AD], OpenSARShip [HLL17], and FUSAR-Ship [HAS20] datasets. We first test the accuracy and efficiency of our methods as compared to other batch active learning methods and sequential active learning. We then test the accuracy of our methods with different embedding techniques on each dataset. Lastly, we look at the impact of data augmentation and the

choice of neural network architecture in transfer learning. It is important to note that our methods use less data than state-of-the-art methods. LocalMax surpasses state-of-the-art accuracy on OpenSARShip, and FUSAR-Ship with 35% and 68% of the data labeled, respectively. The previous state of the art utilized 44% and 70% of the data for OpenSARShip and FUSAR-Ship, respectively [ZZK21].

In our transfer learning experiments, we use PyTorch CNNs pre-trained on the ImageNet dataset to perform image classification on SAR datasets. Unless otherwise stated, we use AlexNet for OpenSARShip and ShuffleNet for FUSAR-Ship since preliminary experiments suggested that they would achieve the best performance among the neural networks tested. The transfer learning results for MSTAR were generally poor and we only present the results with transfer learning from ResNet. These choices are later examined in our comparison between different neural network architectures. Our experiments use both zero-shot and fine-tuned transfer learning. In all the embeddings mentioned, the data is first transformed using the following PyTorch data transformations in order: Resize, CenterCrop, RandomRotation, GaussianBlur, and ColorJitter. Also, LocalMax may not always find batches of the specified size $B = 15$, so it selects up to 15 points in a batch. As evidenced by the later efficiency improvements, we see that this occurs infrequently.

Table 3.1 contains all the parameters used in our experiments. The experiment time measures the time taken to complete the entire active learning process after the core-set selection, including batch selection and model fitting. The time calculation neglects the human labeling time, so the performance enhancements seen in practice will be much larger. Accuracy is measured as the percent of correct predictions by the model in the unlabeled dataset. The source code to reproduce all the results is available [Cal22, CCT20, PVG11]. All experiments were performed in Google Colab with high RAM.

Parameter	Value
Batch size	15
Transfer learning data	5%
Sequential Acquisition Function	Uncertainty

Dataset	Final Labels (%)	TL Architecture
MSTAR	15%	ResNet
OpenSARShip	35%	AlexNet
FUSAR-Ship	68%	ShuffleNet

Table 3.1: Tables of parameters used in our experiments. All experiments use these parameters unless otherwise stated. In the left table, "transfer learning data" refers to the amount of data used in fine-tuned transfer learning. This data is sampled uniformly at random and is then used as part of the core-set before performing DAC. In the right table, "final labels" refers to the size of the labeled dataset as a percent of the total dataset size at the end of the active learning process. Also, "TL architecture" refers to the pretrained PyTorch neural network used for transfer learning on each dataset.

3.4.1 Accuracy And Efficiency

As seen in Table 3.2, LocalMax generally outperforms the other batch active learning methods. Among the batch active learning methods tested, LocalMax attained the highest accuracy with comparable efficiency to the other methods. The time efficiency of LocalMax is slightly worse than random sampling, but this is because random sampling is a very naive approach that achieves much lower accuracy than LocalMax in each dataset. The TopMax method has comparable time and accuracy to LocalMax, but in each test, the accuracy is lower. This is likely because TopMax does not enforce separation of the data in a batch and may select points with lots of shared information. Acq Sample also performs worse than LocalMax by a noticeable amount.

We can see from Table 3.2 that LocalMax has comparable accuracy to sequential active learning, deviating by at most 0.64% on each dataset. Additionally, LocalMax uses an order of magnitude less computational time than sequential active learning. The discrepancy between theoretical efficiency analysis and experimental time efficiency is because human querying time is negligible in these experiments. The code immediately provides ground

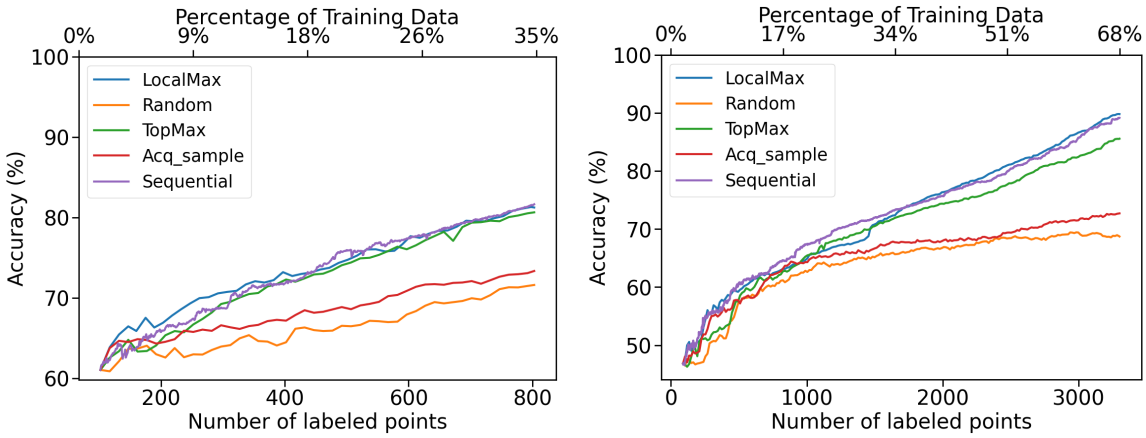
		LocalMax	*Random*	*TopMax*	*Acq_sample*	*Sequential*
Time Consumption	*MSTAR*	26.70s	**24.17s**	25.96s	26.71s	338.11s
	OpenSARShip	5.33s	**4.68s**	4.86s	4.86s	47.43s
	FUSAR-Ship	26.80s	**21.21s**	26.08s	21.55s	322.99s
Accuracy	*MSTAR*	99.69%	92.66%	99.69%	96.27%	**99.93%**
	OpenSARShip	81.25%	71.60%	80.64%	73.34%	**81.65%**
	FUSAR-Ship	**89.83%**	68.73%	85.59%	72.72%	89.19%

Table 3.2: Time consumption and accuracy comparison among different active learning methods. This experiment uses a CNNVAE embedding for MSTAR and zero-shot transfer learning for OpenSARShip and FUSAR-Ship. Additionally, the parameters for all the methods are listed in Table 3.1. LocalMax is the batch sampling method introduced in Section 3.2.2. Random is a batch active learning method that randomly chooses a new batch with the desired size. TopMax is a batch active learning method that chooses the n points with the highest acquisition values. The acq_sample method assigns each point with a probability to be picked proportional to the acquisition value and randomly samples n points as a batch. All batch active learning methods have comparable efficiency and are 9 to 15 times faster than the sequential case. The local max method always achieved higher accuracy than other batch active learning methods and is comparable to the accuracy of sequential active learning.

truth labels when queried, so most of the time in these experiments is observed in the model fit time. In practice, these time differences will better match theoretical predictions where the human query time is the bottleneck. These experiments attest to the efficiency and accuracy of LocalMax relative to other active learning methods.

More detailed plots of the accuracy can be seen in Figure 3.7. These plots show accuracy as a function of the size of the labeled set. In each case, we see a large jump in accuracy with few labels, which is characteristic of the active learning process. After this point, the

accuracy appears to grow linearly with the amount of labeled data. Again, we notice that LocalMax and sequential active learning tend to perform best on both datasets. We can also see that random sampling fails to capture the importance of labeling certain data as its accuracy is much lower throughout the active learning process. This shows that LocalMax is improving the model more substantially than by just increasing the size of the labeled dataset.



(a) OpenSARShip (b) FUSAR-Ship

Figure 3.7: Plots of accuracy v.s. the number of labeled points for five different active learning methods. Details about these active learning methods are shown in the caption of Table 3.2. Panel (a) and (b) contain the results for the OpenSARShip and FUSAR-Ship datasets, respectively. In each panel, our LocalMax method (blue curve) and the sequential active learning (purple curve) are almost identical and are the best-performing methods. According to Table 3.2, LocalMax is much more efficient, proportional to the batch size.

Following the comparison between different batch and sequential active learning methods, we analyze the impact of the acquisition function on LocalMax. The results of these experiments are shown in Figure 3.8, where we see that the uncertainty acquisition function performs best in all experiments. This matches results from previous works on sequential active learning [MMS22]. Figure 3.8 also shows that the uncertainty-based LocalMax beats state-of-the-art in both transfer learning experiments on FUSAR-Ship and OpenSARShip.

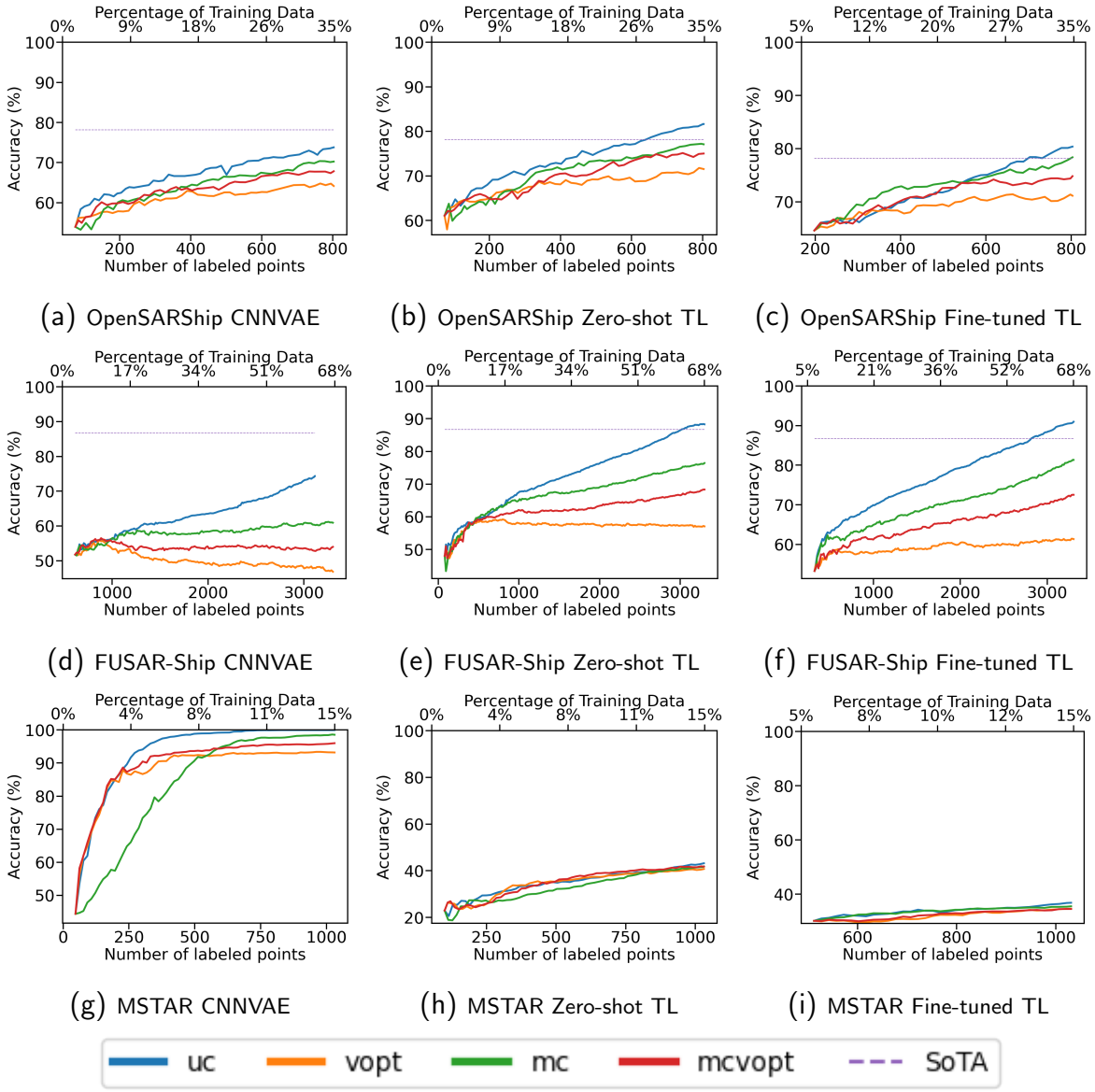


Figure 3.8: Plots of accuracy v.s. Number of labeled points for each embedding and dataset. In each panel, we show four curves generated by LocalMax with acquisition functions (Section 2.3.2), together with the SOTA CNN-based method [ZZK21]. Three rows from top to bottom correspond to the OpenSARShip, FUSAR-Ship, and MSTAR datasets. Three columns from left to right correspond to CNNVAE embedding, zero-shot transfer learning embedding and fine-tuned transfer learning embedding. The UC acquisition function performs best among all the acquisition functions tested. The parameters for these experiments are the same as those specified in Table 3.1.

3.4.2 Sensitivity Analysis

LocalMax Accuracy with Different Embeddings

	State of the Art	Zero-shot	Fine-tuned	Fine-tuned + Augmentation
OpenSARShip	$78.15\% \pm 0.57\%$	$81.02\% \pm 0.76\%$	$79.66\% \pm 0.90\%$	$80.00\% \pm 0.75\%$
FUSAR-Ship	$86.69\% \pm 0.47\%$	$88.57\% \pm 0.35\%$	$91.54\% \pm 3.07\%$	$89.06\% \pm 1.90\%$

Table 3.3: Sample statistics of accuracy after 20 experiments of the batch active learning pipeline with zero-shot transfer learning, fine-tuned transfer learning, and fine-tuned transfer learning with data augmentation (last column). The number in each cell represents the mean $\pm$ one standard deviation across the 20 experiments. The zero-shot and fine-tuned embeddings are the same as mentioned in Section 3.3.1. The parameters in these experiments are the same as those specified in Table 3.1.

Zero-shot Embedding Active Learning Various CNN

	OpenSARShip	FUSAR-Ship
AlexNet	**80.24%**	85.14%
ResNet18	72.14%	87.39%
ShuffleNet	72.34%	88.22%
DenseNet	75.69%	**89.96%**
GoogLeNet	73.28%	86.74%
MobileNet V2	74.15%	86.68%
ResNeXt	76.29%	88.29%
Wide ResNet	73.14%	85.52%

Table 3.4: Accuracy values of one run of LocalMax for different choices of neural networks. Each experiment uses zero-shot transfer learning (no fine-tuning) and the parameter values specified in Table 3.1. The highest accuracy value in each column is bolded. As shown in the table, the range of model performance across architectures is 8.10% and 4.82% for OpenSARShip and FUSAR-Ship, respectively.

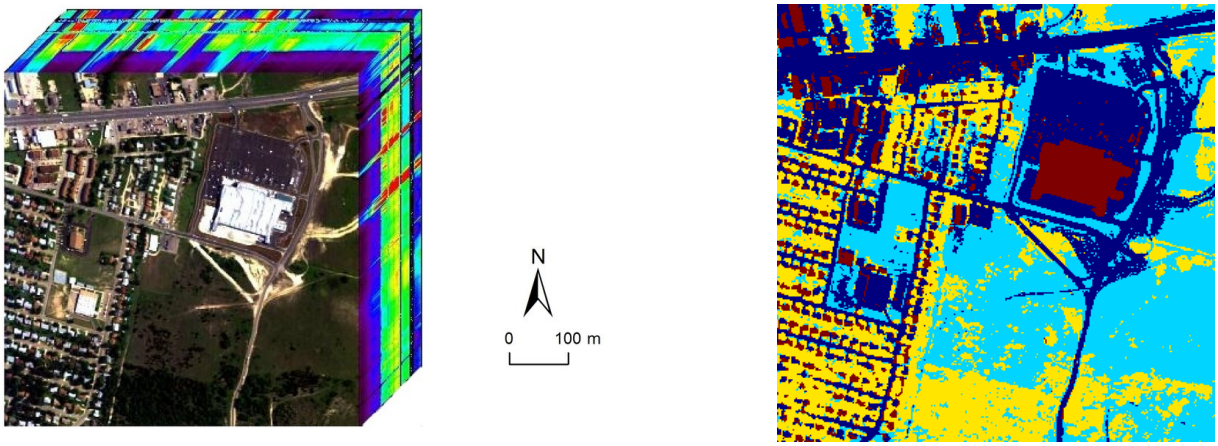
We now look at the sensitivity of our results based on choices in the pipeline. We first look at the impacts of data augmentation and fine-tuning on the final results. Table 3.3 contains summary statistics for an experiment regarding the benefits of using data augmentation and fine-tuning in the transfer learning portion of the pipeline. The results of this experiment are not conclusive between OpenSARShip and FUSAR-Ship. For OpenSARShip, we see that variance is consistently low for the embeddings and zero-shot transfer learning performed best. In contrast, the FUSAR-Ship results had notably higher accuracy and variance for fine-tuned transfer learning without data augmentation. Adding data augmentation reduced variance for fine-tuned transfer learning on both datasets. It is also important to note that each of these experiments showed better accuracy than the previous state-of-the-art. Additionally, zero-shot transfer learning may be the most practical method as it attains comparable performance to the other embeddings, does not require labels in the new dataset, and has no training time.

Lastly, we study the impact of neural network architecture on model performance. Table 3.4 shows the results of one run of LocalMax for different choices of neural network architectures. The range of model performance across architectures is 8.10% for OpenSARShip and 4.82% for FUSAR-Ship. Although this is a large variation, this is a common issue encountered in deep supervised learning. It is important to note that the neural network architectures tested are standard neural networks and weren't designed for SAR data. It is possible that transfer learning from architectures designed for SAR data will experience less variance in the final accuracy [IID21, AMZ18].

3.5 Experiments and Results: MSI and HSI Segmentation

This section shows the experiments and results of our graph-based active learning pipeline on the image segmentation tasks. We run two types of experiments. The first one compares LocalMax, the sequential active learning process, and the other two straightforward batch

sampling approaches for active learning. The second one is the application of our pipeline on image segmentation tasks and the comparison with a similar semi-supervised approach proposed in [MMK17]. Finally, we provide some comments about our experiment results.



(a) raw Image (b) Ground Truth Labels

Figure 3.9: Urban Dataset. Panel(a) shows the raw hyperspectral image we used for experiments. Panel(b) shows the ground-truth labels. Label information: asphalt (navy blue), grass (light blue), trees (yellow), roof (red).

We consider three datasets, Landsat-7, Urban, and Kennedy Space Center (KSC) in the following experiments. Here are brief introductions for these datasets:

1. The Urban dataset was recorded in October 1995 by the Hyperspectral Digital Imagery Collection Experiment (HYDICE) over the urban area in Copperas Cove, TX, U.S. Zhu et al. [ZWX14] provides the ground truth labels with four classes: asphalt, grass, tree, and roof. This dataset contains a hyperspectral image with 307×307 pixels, each corresponding to a 2 square meters area. The raw image includes 210 channels, but we use the clean version with 162 channels after removing some channels due to dense water vapor and atmospheric effects. Figure 3.9 shows the raw image and ground truth labels of the Urban dataset.

2. The Landsat-7 dataset in this paper is a multispectral image of the Colville River

(Alaska, USA) from the RiverPIXELS dataset [SR22], which provides paired Landsat and water-and-sediment labeled patches of size $256 \times 256 \times 6$, where the 6 multispectral channels correspond to Blue, Green, Red, Near IR, Shortwave IR 1, and Shortwave IR 2. Each pixel in the image covers roughly 900 m^2. The RiverPIXELS provides ground-truth labels of this image with three classes, land, water, and bare sediment.

3. The Kennedy Space Center Dataset is a hyperspectral image at the Kennedy Space Center (KSC) in Florida, acquired by the NASA AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) instrument. This hyperspectral image has size 512×614. The raw image includes 224 channels, while we are using the clean version with 176 channels after removing water absorption and low SNR channels. There are 314368 pixels in this dataset, while only 5211 (around 1.66%) of them have ground-truth labels. The ground-truth labels include 13 classes of different land coverings in this region.

We consider the overall accuracy (OA) to evaluate the performance of different methods in this Section. For a multi- or hyperspectral image $I \in \mathbb{R}^{N_1 \times N_2 \times c}$, let $y_{i,j}$ and $y_{i,j}^{\dagger}$ are the predicted label and the ground-truth label of the pixel indexed by (i, j), $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$. The definition of overall accuracy is:

$$\text{OA} = \frac{\text{Number of Correctly Classified Pixels}}{\text{Total Number of Pixels}} = \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \delta(y_{i,j}, y_{i,j}^{\dagger})}{N_1 N_2}, \tag{3.12}$$

where the δ-function is defined by:

$$\delta(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{3.13}$$

3.5.1 Comparison between LocalMax and Sequential Active Learning: Accuracy and Efficiency

In this section, we conduct our experiments solely on the Urban dataset. We utilize the hyperspectral pixel values as the feature vector, meaning that the corresponding feature

vector consists of 162 channels for each pixel. We use four acquisition functions to sample up to 134 pixels (0.15% of all pixels) with various active learning sampling methods. We initialize the labeled set with 10 randomly selected pixels from each class (a total of 40 pixels) and sample an additional 94 pixels based on the active learning methods. For a given acquisition function $\mathcal{A}$, we consider four sampling methods: Sequential, Random, Top-Max, and LocalMax, with the last three being batch active learning methods with a batch size of B.

- **Sequential sampling** selects the global maximum node of $\mathcal{A}$ to update the current labeled set. The query set $\mathcal{Q} = \{\mathbf{x}^*\}$ and $\mathbf{x}^* = \arg\max_{\mathbf{x}\in X_u} \mathcal{A}(\mathbf{x})$.
- **Random sampling** selects a batch of B unlabeled nodes according to a uniform distribution over the unlabeled node set X_u.
- **Top-Max sampling** selects a batch of B unlabeled nodes as the top-B maximum of $\mathcal{A}$, i.e., the query set $\mathcal{Q} = \{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_B}\} \subset X_u$ where $\mathbf{x}_{i_1} = \arg\max_{\mathbf{x}\in X_u} \mathcal{A}(\mathbf{x})$ and $\mathbf{x}_{i_b} = \arg\max_{\mathbf{x}\in X_u \setminus \{\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_{b-1}}\}} \mathcal{A}(\mathbf{x})$ for $b = 2, 3, \ldots, B$.
- **LocalMax sampling** is the method we proposed in Section 3.2.2.

Figure 3.10 illustrates the relationship between accuracy and the number of labeled pixels for the four acquisition functions, while Table 3.5 presents the time consumption and accuracy values for label rates of 0.1% and 0.15%. From these experiments, we draw the following conclusions:

1. **Accuracy Performance:** Sequential active learning demonstrates the highest accuracy performance, as shown in Figure 3.10, which depicts its superior accuracy for nearly all numbers of labeled pixels. Our batch active learning method, LocalMax, achieves the second-best accuracy values and performs almost identically to the sequential method, particularly for larger numbers of labeled pixels (i.e., when the number of

labeled pixels are greater than 80). This is further supported by Table 3.5, highlighting the top-2 accuracy values in bold. LocalMax consistently exhibits accuracies in the top 2 and occasionally outperforms Sequential.

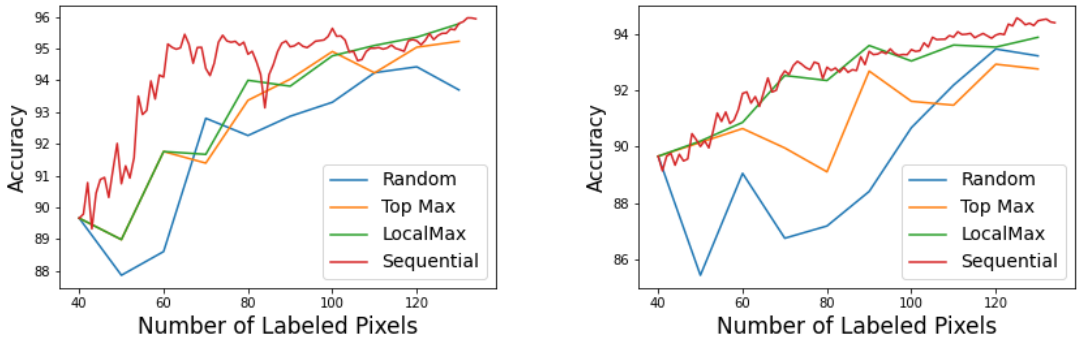
2. **Efficiency Performance:** According to the timings in Table 3.5, LocalMax requires approximately the same amount of time as the Random and Top-Max sampling methods. At the same time, Sequential active learning takes around eight times longer. This time multiplier of 8 is close to the theoretical multiplier of 10 (equal to the batch size B), as discussed in Section 3.2.2.

In summary, LocalMax batch active learning is significantly more efficient than Sequential active learning without substantially compromising accuracy.

3.5.2 Semi-Supervised Image Segmentation with Low Label Rates

We perform segmentation experiments on three datasets: Landsat-7, Urban, and KSC. The performance is evaluated based on the overall accuracy as a function of the amount of labeled data used in training. Our method is compared with the graph-based semi-supervised method, abbreviated as **GL-SSL**, proposed by Meng et al. [MMK17]. There are several differences between our graph learning method and GL-SSL. Firstly, our method uses the KNN approach to build a sparse similarity graph, while GL-SSL is based on a fully connected graph and uses the Nyström extension method to approximate the graph Laplacian matrix (Section 2.2.1). Secondly, our approach solves the optimal node function by minimizing the energy function of graph Laplace learning, while GL-SSL minimizes a regularized Ginzburg-Landau functional [GMB14, MGB14, MKB13]. We have re-implemented the GL-SSL code[3] in Python, and the corresponding results are based on our implementation. The parameters selected for GL-SSL are based on their recommendations with some fine-tuning. To avoid bloated content, we only provide results of GL-SSL with the randomly selected training

[3]http://www.ipol.im/pub/art/2017/204/



(a) Uncertainty Acquisition Function

(b) MC Acquisition Function

(c) MCVOpt Acquisition Function

(d) VOpt Acquisition Function

Figure 3.10: Comparison between batch and sequential active learning methods for four acquisition functions. Each panel includes four curves, of which the X-axis is the number of labeled pixels and the Y-axis is the accuracy. The blue, yellow, green, and red curves correspond to the Random, Top-Max, LocalMax, and Sequential sampling method respectively for the active learning process. More details on accuracy values and time consumption are shown in Table 3.5. Descriptions of each sampling method are in Section 3.5.1.

dataset. We also test the performance of GL-SSL with the training set selected by our active learning approach. Since the active learning process introduced in this paper is designed for graph Laplace learning, the overall accuracy performance of GL-SSL with an active learning training set is almost the same as that of a random training set. Our method's overall accuracy (OA) result or GL-SSL with the randomly sampled training dataset is the average

Comparison between different Active Learning Sampling Methods

Label Percentage			0.1%		0.15%	
Acq	Sampling	Batch Size	Time (s)	Acc (%)	Time (s)	Acc (%)
UC	Sequential	1	1290.13	**94.90**	2576.86	**95.93**
	Random	10	180.42	92.86	358.84	93.69
	Top-Max	10	169.59	**94.03**	343.24	95.22
	LocalMax	10	165.30	93.81	326.56	**95.78**
MC	Sequential	1	1250.22	**92.70**	2505.60	**94.40**
	Random	10	166.59	88.41	327.95	93.22
	Top-Max	10	165.07	92.68	327.35	92.71
	LocalMax	10	165.28	**93.59**	334.10	**93.88**
MCVOpt	Sequential	1	1131.92	**93.70**	2257.06	**94.99**
	Random	10	161.83	91.34	322.44	92.02
	Top-Max	10	160.61	88.49	323.99	87.02
	LocalMax	10	164.79	**93.55**	321.92	**95.27**
VOpt	Sequential	1	1289.92	**92.60**	2576.86	**94.54**
	Random	10	175.99	86.80	346.61	91.91
	Top-Max	10	169.91	88.75	339.96	85.62
	LocalMax	10	175.61	**90.24**	357.60	**93.91**

Table 3.5: This table shows the efficiency and accuracy performance of active learning sampling methods with different acquisition functions. The 'Acq' column refers to the acquisition function. The 'Sampling' column refers to the choice of active learning sampling methods, including Sequential, Random, Top-Max, and LocalMax, the last three of which are batch active learning. The 'B' column is the batch size. Timings and accuracies are shown for up to 0.1% and 0.15% labeled pixels in the Urban dataset. The top two accuracy values are bolded for each acquisition function. Descriptions of each sampling method are in Section 3.5.1.

OA value of 15 random samples. Each time, the random sampling process starts with 1 random pixel in each class and then randomly chooses the rest of the labeled pixels with equal probability.

The results for the **Landsat-7 dataset** are presented in Table 3.6 and Figure 3.11. We construct the similarity graph by using each pixel's non-local means feature vector. The neighborhood patch size is set to 7×7, resulting in a 294-dimensional feature vector for each pixel. We sample up to 200 pixels (0.3% of all pixels) based on the random initialization of one pixel in each class (three pixels in total for the initialization) using the LocalMax batch active learning approach with a batch size of 20. Additionally, we sample up to 3300 labeled pixels (approximately 5% of all pixels) based on the random initialization of 10 labeled pixels in each class (30 in total) and a batch size of 100. According to the overall accuracy values shown in Table 3.6, the UC and MCVOpt acquisition functions achieve better accuracy with only 0.3% labeled pixels than randomly selecting 5% labeled pixels. Figure 3.11 illustrates the segmentation result of our graph-based batch active learning method with UC acquisition functions.

The results for the **Urban dataset** are presented in Table 3.7 and Figure 3.12. In these experiments, we use the hyperspectral pixel values as the feature vector for each pixel. We sample up to 286 pixels (0.3% of all pixels) based on the random initialization of one pixel in each class (four pixels in total for the initialization) using LocalMax with a batch size of 10. Additionally, we sample 4700 pixels (approximately 5% of all pixels) using LocalMax with a batch size of 100, based on the random initialization of 10 pixels in each class (40 in total). It is evident from the results that LocalMax batch active learning with the uncertainty (UC) acquisition function achieves an accuracy of 97.30% with only 0.3% labeled pixels, which is comparable to the accuracy of 97.76% obtained with 10% randomly selected labeled pixels.

For the **KSC dataset**, we only consider and calculate our results on the 5211 labeled pixels (approximately 1.66%) out of the total 314368 pixels, as visualized in Figure 3.13. According to the ground-truth labels, we segment this hyperspectral image into 13 classes.

We sample up to 325 pixels (6% of all pixels with ground-truth labels) based on the random initialization of 1 pixel in each class (13 pixels in total for the initialization). Table 3.8 presents the overall accuracy of the KSC dataset, demonstrating that our LocalMax batch active learning with the uncertainty (UC) acquisition function performs best. Figure 3.13 illustrates the segmentation result of our graph-based batch active learning method with the UC acquisition function.

In summary, with a very low label rate (less than 0.5%), our method has a relatively good performance on the overall accuracy, while GL-SSL doesn't perform well. Our graph-based active learning approach can significantly reduce the number of labeled pixels required for the semi-supervised image segmentation task. According to Tables 3.6 and 3.7, the graph Laplace learning with 0.3% labeled pixels selected by active learning has a similar accuracy performance to that with 5% to 10% randomly selected labeled pixels.

3.5.3 Comments on Our Experiments

Our experiments in Sections 3.5.1 and 3.5.2 provide valuable insights into the performance of different acquisition functions and the LocalMax sampling method. When the percentage of training data is relatively small, the MCVOpt acquisition function exhibits similar or even better performance compared to the UC acquisition function. This observation is particularly evident in Table 3.6, where the UC achieves only 25% accuracy with 0.1% labeled data, while other acquisition functions surpass 90% accuracy. However, as the percentage of training data increases, the UC acquisition function consistently outperforms the others regarding overall accuracy. This phenomenon can be attributed to the different tendencies of the acquisition functions towards exploration or exploitation in the active learning process. The UC acquisition function primarily focuses on exploiting the classifier's current decision boundaries based on the current labeled set by querying data points along the boundaries between different classes. In contrast, the other three acquisition functions, namely MC, VOpt, and MCVOpt, prioritize exploration over exploitation, as they are designed to explore the

Overall Accuracy of a Landsat-7 Multispectral Image (65536 pixels)

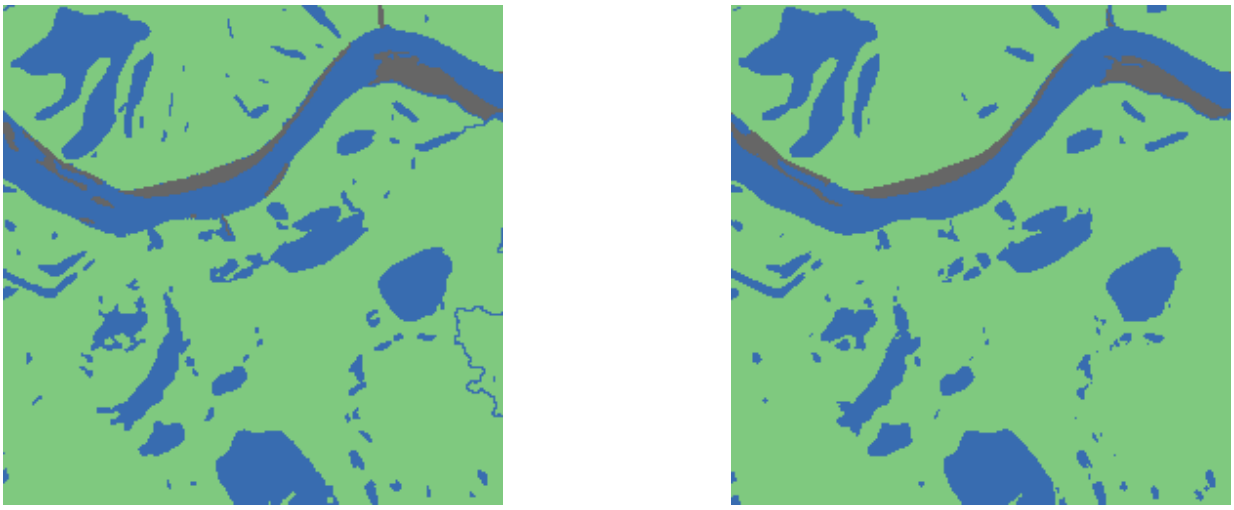
Method	***Sampling***	***Label Rate (Batch Size)***			
		0.1%(20)	*0.2%(20)*	*0.3%(20)*	*5%(100)*
Ours	LM UC	25.17%	**95.92%**	96.23%	**98.65%**
Ours	LM MC	91.75%	95.04%	95.80%	97.50%
Ours	LM MCVOpt	**94.81%**	95.23%	**96.25%**	97.74%
Ours	LM VOpt	93.61%	94.4%	94.78%	96.44%
Ours	Random	88.43%	91.38%	92.55%	96.12%
GL-SSL [MMK17]	Random	20.77%	41.44%	48.56%	94.94%

Table 3.6: The overall accuracy (averaged over 15 random samples) of our LocalMax batch active learning method with different acquisition functions and sampling strategies on the Landsat-7 multispectral image. Results are shown for various label rates, with each process initiated using 10 labeled pixels per class (30 total).

Overall Accuracy of Urban Dataset (94249 pixels)

Method	***Sampling***	***Label Rate (Batch Size)***				
		0.1%(10)	*0.2%(10)*	*0.3%(10)*	*5%(100)*	*10%*
Ours	LM UC	**94.96%**	**95.56%**	**97.30%**	**99.71%**	N/A
Ours	LM MC	90.71%	93.31%	94.84%	98.60%	N/A
Ours	LM MCVOpt	94.13%	95.33%	95.35%	98.94%	N/A
Ours	LM VOpt	91.32%	94.27%	94.52%	97.44%	N/A
Ours	Random	87.20%	91.93%	93.30%	97.20%	**97.76%**
GL-SSL [MMK17]	Random	53.06%	55.08%	58.79%	90.71%	93.48%

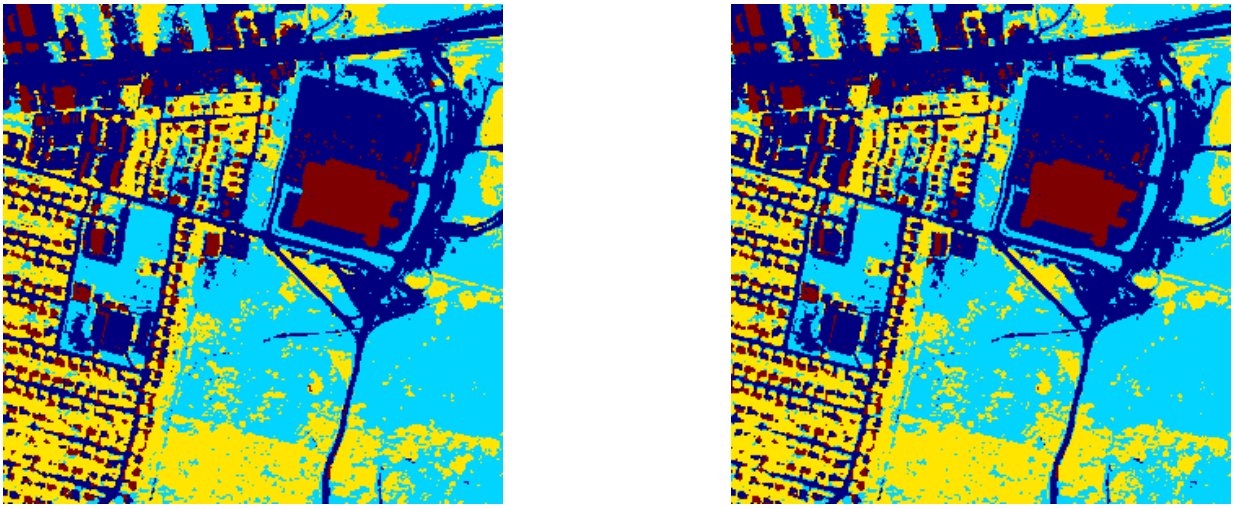
Table 3.7: The overall accuracy (averaged over 15 random samples) of our LocalMax batch active learning method with different acquisition functions and sampling strategies on the Urban dataset. Results are shown for various label rates, with each process initiated using 1 labeled pixel per class (3 total).



(a) Ground-truth Labels

(b) OA: 96.23%

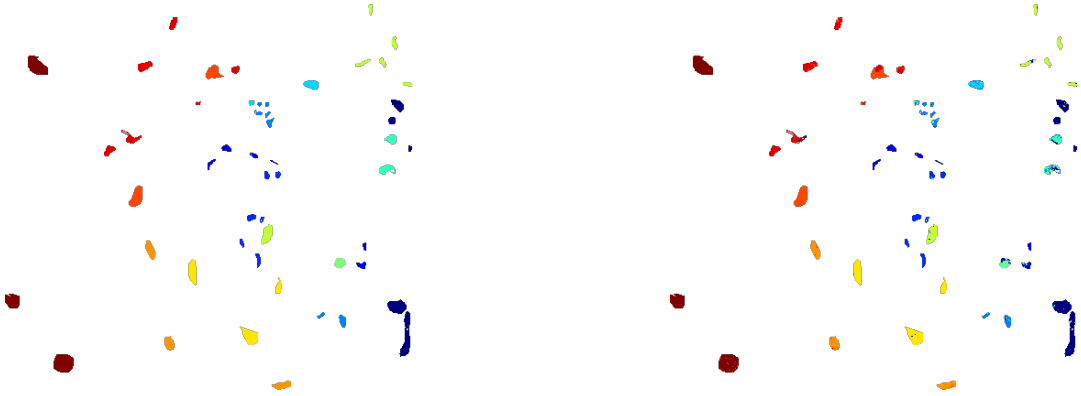
Figure 3.11: The ground-truth labels and segmentation results of a Landsat-7 multispectral image from the RiverPIXELS dataset. The segmentation was performed using 0.3% labeled pixels sampled with the LocalMax batch active learning method, a batch size of 20, and the UC acquisition function, with feature vectors of 7×7 neighborhood patches.



(a) OA: 97.30%

(b) OA: 95.35%

Figure 3.12: The segmentation result of the Urban dataset with 0.3% labeled pixels sampled according to LocalMax batch active learning with batch size 10 with different acquisition functions, (a): UC; (b): MCVOpt. Label information: asphalt (navy blue), grass (light blue), trees (yellow), roof (red). The ground-truth labels are in Figure 3.9.



(a) Ground-truth labels

(b) OA: 89.83%

Figure 3.13: The ground-truth (a) of 5211 pixels and segmentation result (b) of the KSC dataset. The segmentation result is with 6% labeled pixels sampled according to LocalMax batch active learning with batch size 10 and the UC acquisition function.

Overall Accuracy of KSC Dataset (5122 pixels with ground-truth labels)

Method	***Sampling***	***Label Rate (Batch Size)***		
		2.5%(5)	*4%(5)*	*6%(5)*
Ours	LocalMax UC	**85.47%**	**88.66%**	**89.83%**
Ours	LocalMax MC	82.04%	85.86%	87.85%
Ours	LocalMax MCVOpt	85.12%	85.86%	87.13%
Ours	LocalMax VOpt	83.22%	84.68%	86.66%
Ours	Random	82.22%	85.22%	85.83%
GL-SSL [MMK17]	Random	<u>80.37%</u>	81.67%	83.55%

Table 3.8: The overall accuracy (averaged over 15 random samples, based on 5211 labeled pixels) of our LocalMax batch active learning method with different acquisition functions and sampling strategies on the KSC dataset. Results are shown for various label rates, with each process initiated using 1 labeled pixel per class (13 total). The underlined value is copied from paper [MMK17].

geometric structure of the entire dataset. At the initial stages of the active learning process, exploring the dataset's inherent geometric or clustering structure is crucial. However, when the labeled percentage or the current accuracy is relatively high, exploiting the decision boundary becomes more beneficial for further improving the overall classification accuracy. Based on our empirical findings, we recommend using the UC acquisition function, but we also caution that it may not always be the optimal choice due to its tendency to prioritize exploitation during the active learning process.

Furthermore, Figure 3.10 reveals a performance gap between LocalMax and sequential active learning when the number of labeled nodes is small. The primary reason for this initial gap is that a newly labeled node can significantly influence the acquisition function's value when the amount of labeled data is limited. In such circumstances, LocalMax may not perform optimally since the next global maximum of the acquisition function might not be close to any local maximum of the current acquisition function. However, it is important to note that this gap does not necessarily persist throughout the entire active learning process. As the number of labeled data increases, the LocalMax batch sampling method can better approximate sequential sampling, leading to improved performance. Overall, our experiments demonstrate the effectiveness of the proposed LocalMax sampling method and provide guidance on the selection of acquisition functions for active learning in hyperspectral image segmentation tasks.

3.6 Conclusion

In this chapter, we present two novel batch active learning methods, DAC for core-set selection and LocalMax for batch sampling, and their applications in SAR image classification and multi- and hyperspectral image segmentation. Our proposed methods demonstrate excellent performance in both domains, achieving high accuracy with a limited amount of labeled data while significantly improving efficiency compared to sequential active learning

methods.

In the domain of SAR image classification, we apply neural network embedding techniques (CNNVAE and transfer learning) before the graph learning and active learning process. Our approach outperforms state-of-the-art SAR classification methods on the OpenSARShip and FUSAR-Ship datasets. Moreover, our methods significantly improve efficiency compared to sequential methods while maintaining comparable accuracy, and attain higher accuracy than other common batch active learning methods on the tested datasets.

We also apply the LocalMax batch active learning method to multi- and hyperspectral image segmentation. Our graph-based batch active learning pipeline performs better than a similar graph-based method proposed in [MMK17]. Our approach requires fewer labeled pixels to achieve better overall accuracy, highlighting the benefits of carefully selecting points to label through active learning in graph-based semi-supervised classification. Experiments demonstrate that LocalMax not only accelerates the sampling process but also maintains accuracies similar to sequential active learning using the same acquisition function.

The success of our DAC and LocalMax methods in both SAR image classification and multi- and hyperspectral image segmentation demonstrates the versatility and effectiveness of our batch active learning approach. By leveraging the strengths of active learning, graph-based methods, and efficient batch selection, we develop a powerful framework for semi-supervised learning in various image analysis tasks. These advancements contribute to reducing the reliance on large amounts of labeled data, which is often costly and time-consuming to acquire, while still achieving high-quality results.

CHAPTER 4

Graph-based Active Learning for Surface Water and Sediment Detection in Multispectral Images

This chapter reuses materials from the author's publications [CMB23b, CMB24]. IEEE copyrighted material from [CMB23b] is reused in this chapter, with the approval of the senior author Andrea L. Bertozzi and following the requirements outlined by IEEE for thesis/dissertation reuse.[1] *The work [CMB24] is submitted to IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.*

With the continuous advancements in remote sensing technology, high-resolution multispectral satellite imagery has become increasingly abundant, providing a crucial data source for monitoring surface water bodies and river sediments. Traditional methods for water and sediment extraction, such as thresholding and decision trees [Gao96, McF96], struggle to fully utilize the rich information contained in multispectral imagery and often require a large amount of manually labeled data for training, leading to low efficiency. In recent years, deep learning methods like Convolutional Neural Networks (CNNs) [LBD89] have achieved remarkable results in remote sensing image analysis [ZTM17, MLZ19]. However, they still face challenges such as insufficient labeled data and limited model generalization ability.

To overcome these difficulties, we propose a Graph Active Learning Pipeline (GAP) for extracting surface water and sediment from multispectral remote sensing images [CMB23b]. This method employs an active learning strategy to optimize the selection of a small number

[1] ©2023 IEEE. Reprinted, with permission, from [CMB23b]

of representative pixels for manual labeling during the training process, thereby constructing an efficient training set. Experimental results demonstrate that our method, using only 3,270 training samples, achieves higher accuracy than CNN-based methods trained on 2.1 million samples [IBP19]. This clearly proves the superiority of the GAP method in reducing labeling costs and improving classification accuracy.

Building upon the GAP method, we further propose an enhanced Contrastive Graph-based Active Learning Pipeline (CGAP) [CMB24]. CGAP introduces a feature-embedding neural network based on contrastive learning [CKN20, CKS20, KTW20] into the GAP framework, mapping high-dimensional raw features to a lower-dimensional space to facilitate more efficient graph learning. Additionally, we design specialized data augmentation strategies to improve the robustness of the embedded features against geometric transformations, resolution variations, and light cloud cover. Moreover, we develop a Python package that seamlessly integrates Google Earth Engine with CGAP, providing a user-friendly solution for rapid and accurate environmental predictions on a global scale.

Our work in this chapter makes the following innovations and contributions:

1. We propose pipelines based on graph-based active learning that significantly reduces manual labeling costs and improves the accuracy of water and sediment detection;

2. We introduce contrastive learning strategies to enhance graph learning efficiency and robustness, expanding the applicability of the method;

3. We develop an easy-to-use software package that promotes the application of our pipelines in global environmental monitoring.

4.1 Introduction

Mapping surface water dynamics is crucial for a host of environmental, engineering, and management problems, including climate studies, flood monitoring and mitigation, freshwa-

ter resource management, water quality analyses, and earth science research [BNA13, CRS21, CSK21, KPR21]. 80% of the Earth's population is under high levels of threat to water security [VMG10]. In tandem with addressing these needs, opportunities for understanding surface water dynamics are becoming more abundant with the rise of global, remotely sensed surface water observations [SPR20], but new technologies are needed to fully exploit these data archives.

Rivers are at the center of water security, providing freshwater and ecologic resources that support agriculture and human development worldwide. Many rivers are highly dynamic to environmental conditions [DCS17, RNC14], and studying these dynamics from remotely-sensed images is an active area of research in Earth Sciences [SKF17]. The importance of automated surface water detection from remotely-sensed images is highlighted by significant efforts that have published global datasets or pre-trained models of surface water.

Global surface water datasets and models are effective at capturing the majority of surface water, yet often lack the local precision required for specific applications such as measuring river widths [ASP13, LPA20] or estimating river migration rates [RSP16, SKF17], where the targeted detail may be as narrow as 1-2 pixels along the river boundary. To address this, researchers typically develop local models, which necessitates the labor-intensive task of manual training data labeling, potentially compromising model accuracy. For some river studies, an additional class representing in-channel sediment or highly-turbid water may be desired [SKF17, LWW11]. In our case, we aim to identify rivers at their so-called "bankfull" state [Bje07], which we define as the union of water and active (unvegetated) in-channel sediment bars [SKF17]. Schwenk et al. created a high-quality hand-labeled dataset, RiverPIXELS, [SR22] consisting of labeled water and in-river, unvegetated sediment from Landsat multispectral images. We want to build machine learning models on RiverPIXELS to detect surface water and sediment pixels globally in multispectral satellite images.

There are some published datasets or models that are directly applicable to the surface water detection task. The Global Surface Water dataset [PCG16], which aimed to find all

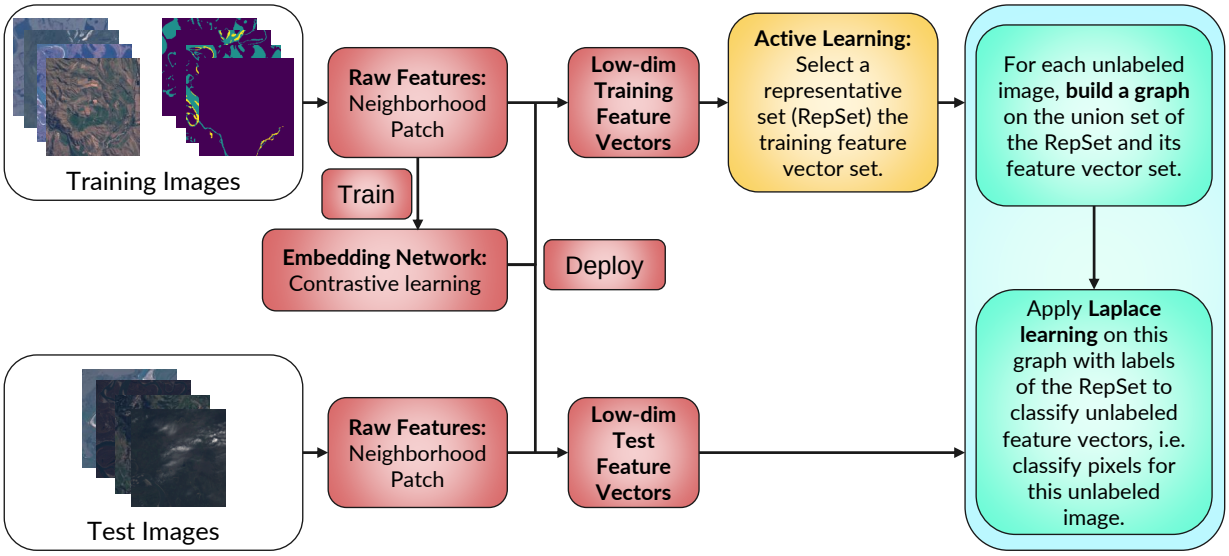


Figure 4.1: The flowchart of our basic contrastive graph-based active learning pipeline (CGAP): 1. (Red Boxes) Use a neural network trained by contrastive learning to preprocess images into feature vectors. 2. (Yellow Box) Condense the labeled feature vector set into a smaller representative set (RepSet) using active learning approaches. 3. (Cyan Box) Build a graph based on the union of the RepSet and the unlabeled feature set. Then, apply graph learning approaches to predict labels for unlabeled features.

water pixels in the Landsat archive, trained an SVM model followed by post-processing to account for confounding features such as ice, snow, volcanic ash, etc. The USGS continues to improve on their Dynamic Surface Water Extent product [Jon19], which provides a surface water map for most Landsat images based on spectral mixing methods. In addition to published datasets, pre-trained models have also been published. Of note is Deepwatermap [IBP19], a convolutional neural network (CNN) U-net model trained using a large collection of publicly available datasets.

The identification of surface water from satellite images can be treated as an image segmentation problem, wherein labels are given to each pixel in an image such that pixels with the same label share certain characteristics. Image segmentation has been approached from

many angles. There are partial differential equation (PDE)-based methods [MS89, KWT88, CV01] for unsupervised segmentation, and deep learning methods like U-Net [RFB15] for supervised segmentation on extensive annotated training datasets.

The RiverPIXELS dataset comprises only 104 images, which motivates us to explore semi-supervised methods like graph learning. As a related example, graph convolutional networks (GCN) have been utilized for wetland classification, outperforming CNN models [JMG22]. Graph-based learning approaches for image segmentation typically involve constructing a similarity graph based on pixel features, where each pixel's feature vector serves as a node, and the edge weights represent the similarity between nodes. However, constructing a graph on millions of pixels, such as those in the 104 images of the RiverPIXELS dataset, can be computationally inefficient. To address this challenge, we propose GAP [CMB23b], which employs an active learning approach [Set09, MMS22] to select representative samples from the training set, avoiding the need to construct a graph on the entire dataset.

In addition to GAP, we present CGAP as the focus of this chapter, which is a significant enhancement to GAP that further improves its stability and efficiency. In fact, the graph learning and active learning parts in both GAP and CGAP are the same. The only difference between these two pipelines is the feature processing. Instead of using raw neighborhood patches as feature vectors for graph construction, CGAP preprocesses them through a shallow feature-embedding neural network. This network is trained using the contrastive learning approach [CKN20, KTW20], which has shown impressive results in SAR image classification when combined with graph learning [BOC23]. CGAP has two versions: a basic version (B-CGAP) and an adaptive version. The flowchart of B-CGAP is illustrated in Figure 4.1.

The details and experimental results presented in this chapter can be fully replicated using the code available in our GitHub repositories of GAP[2], and CGAP[3]. For a quick start

[2] `https://github.com/wispcarey/SurfWater-Graph-Active-Learning-GAP-`

[3] `https://github.com/wispcarey/CGAP-SurfaceWaterDetection`

with our tool, you can directly visit the GitHub repository of GraphRiverClassifier (GRC)[4].

4.2 Graph-based Active Learning Pipeline

This section introduces our proposed graph-based active learning pipelines for detecting surface water and sediment pixels in multispectral images, including the original GAP [CMB23b], and CGAP with the contrastive embedding feature vectors. We provide the basic CGAP (B-CGAP) for the training process based on the RiverPIXELS dataset [SR22], which is considered fully labeled. In addition, we developed the adaptive CGAP (A-CGAP) for potential extra unlabeled data, which should be labeled in a human-in-the-loop process under the guidance of active learning.

Consider the training image set $\mathcal{I} = \{I_i\}_{i=1}^{n_l}$ and the test image set $\tilde{\mathcal{I}} = \{\tilde{I}_j\}_{j=1}^{n_u}$. In training via the B-CGAP on $\mathcal{I}$, it is assumed that all ground-truth labels of pixels in $\mathcal{I}$ are available. In contrast, in training via the A-CGAP, one does not have to a priori know any labels in $\mathcal{I}$.

In both pipelines, we first preprocess the images into feature vectors, each of which corresponds to a pixel, by training a feature-embedding neural network by contrastive learning. Then we sample a training feature vector set, termed the representative set (RepSet) $\mathcal{R}$, as a subset of the whole preprocessed feature set $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$. Each feature vector in $\mathcal{R}$ has its ground-truth label available (for A-CGAP, it should be obtained by human experts).

The core idea of our pipeline is to then use this RepSet for classifying the pixels in each of the test set images in $\tilde{\mathcal{I}}$. For each test set image $\tilde{I}_j \in \tilde{\mathcal{I}}$ with N_0 pixels and corresponding unlabeled feature vector set $\tilde{X}_j = \{\tilde{\mathbf{x}}_1^j, \tilde{\mathbf{x}}_2^j, \ldots, \tilde{\mathbf{x}}_{N_0}^j\}$, we generate a weighted graph $G = (\mathcal{R} \cup \tilde{X}, W)$ according to Section 2.2.1 and then apply Laplace learning (Section 2.2.2) to classify those unlabeled feature vectors in $\tilde{X}_j$.

[4] `https://github.com/wispcarey/GraphRiverClassifier`

4.2.1 Contrastive Learning

Contrastive Learning has emerged as a powerful technique in the field of deep learning, particularly for learning visual representations without extensive labeled datasets. It is a training strategy that can be applied to various types of neural networks. At its core, contrastive learning aims to learn embeddings by maximizing the similarity between augmented views of the same data point while minimizing the similarity between embeddings of different data points.

Chen et al. introduced the SimCLR framework, which simplifies the contrastive learning paradigm by eliminating the need for specialized architectures or a memory bank [CKN20]. This framework was further expanded in their subsequent work, demonstrating the effectiveness of large-scale self-supervised learning for improving semi-supervised learning performance [CKS20]. For a neural network f, in a minibatch of m data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, each $\mathbf{x}_k$ is augmented into $\tilde{\mathbf{x}}_{2k-1}, \tilde{\mathbf{x}}_{2k}$ and process through the neural network $\mathbf{z}_{2k-1} = f(\tilde{\mathbf{x}}_{2k-1}), \mathbf{z}_{2k} = f(\tilde{\mathbf{x}}_{2k})$. The SimCLR loss is defined by

$$\begin{aligned} \mathcal{L} &= \frac{1}{2m}\sum_{k=1}^{m}[\ell(2k, 2k-1) + \ell(2k-1, 2k)], \\ \ell(i,j) &= -\log \frac{\exp(g(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k\neq i}^{2m} \exp(g(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \end{aligned} \tag{4.1}$$

where τ is a constant parameter, and $g(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$ is the angular (cosine) similarity.

Khosla et al. [KTW20] extend SimCLR from self-supervised learning into a supervised version (SupCon) by leveraging label information. SupCon enhances the discriminative power of the embeddings by encouraging embeddings from the same class to cluster together, significantly outperforming traditional cross-entropy loss for supervised learning in many cases. In the supervised setting, we have the ground-truth labels $\{y_1, y_2, \ldots, y_{2m}\}$, $y_{2k-1} =$

y_{2k} for $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{2m}\}$. Based on (4.1), the supervised contrastive loss is defined by

$$\begin{aligned}\mathcal{L}^{\text{sup}} &= \sum_{i=1}^{2m} \frac{-1}{|P(i)|} \sum_{j \in P(i)} \log \frac{\exp(g(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k\neq i}^{2m} \exp(g(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \\ &= \sum_{i=1}^{2m} \frac{1}{|P(i)|} \sum_{j \in P(i)} \ell(i, j),\end{aligned} \tag{4.2}$$

where $P(i) = \{k \neq i : y_k = y_i\}$ is the set of indices with the same label of $\mathbf{z}_i$. The SimCLR loss (4.1) only considers augmented pairs $\mathbf{z}_{2k-1}, \mathbf{z}_{2k}$ while the SupCon loss (4.2) considers all pairs with the same label in the minibatch. In this paper, we set $\tau = 0.5$ for both the SimCLR loss (4.1) and the SupCon loss (4.2).

We utilize the angular distance as a similarity measure in both the contrastive learning approach and the graph construction methodology presented in Section 2.2.1. This deliberate choice facilitates the natural application of graph-based Laplace learning as a classifier for handling the embedding features produced by contrastive learning.

4.2.2 Feature Preprocessing

For every image I within the set $\mathcal{I} \cup \tilde{\mathcal{I}}$, we extract a neighborhood patch centered around each pixel to form raw feature vectors, considering all N_0 pixels in the image. The raw feature cube for each pixel is of the size $(2k+1) \times (2k+1) \times C$. The number of channels for multispectral images in the RiverPIXELS dataset is $C = 6$. Denote the set of raw feature cubes of I by $\{\mathbf{x}_1^{\text{raw}}(I), \mathbf{x}_2^{\text{raw}}(I), \ldots, \mathbf{x}_{N_0}^{\text{raw}}(I)\}$.

For our **GAP method**, we apply the non-local means feature vector [BCM05], referring to Section 3.3.2 for more details. The extracted feature vector of each pixel has the size $6(2k+1)^2$, where k is the neighborhood patch marginal parameter. Practically, in GAP, we choose $k = 3$, and the output feature has the size 294.

For our **CGAP method**, we apply contrastive learning to train a feature embedding neural network to process the raw feature patches $\{\mathbf{x}_i^{\text{raw}}\}_{i=1}^{N_0}$, each of which is with the size $9 \times 9 \times 6$ corresponding to the neighborhood patch marginal parameter $k = 4$. Before

training the feature embedding neural network, we design some transformations for those neighborhood patches:

1. **Horizontal Flip**: Horizontaly flip the patch.
2. **Vertical Flip**: Verticaly flip the patch.
3. **Rotation**: Select a random angle and rotate the patch clockwise. Then crop the output into 9×9. This might introduce some zero values in the output patch.
4. **RandomPixelAugmentation**: Randomly select some pixels and set their values to 1 for all six channels.
5. **CenterCropResize**: Crop the image from the center based on a random scale and then resize the image to 9×9.
6. **Gaussian Reweight**: Reweight the patch by a 9×9 Gaussian kernel matrix.

Each one of these six transformations is applied simultaneously to each channel of the patch. The final random augmentation design for contrastive learning consists of a sequential combination of these six transformations, where the first five are applied with a 50% probability of being executed, while the last transformation, **Gaussian Reweight**, is guaranteed to be applied.

The first three transformations, **Horizontal Flip**, **Vertical Flip**, and **Rotation**, are designed for the robustness of geometry transformations. The fourth one, **RandomPixelAugmentation**, is designed for the possible light coverage of the cloud. The fifth one, **CenterCropResize**, is designed for the robustness of different (higher) resolutions. The last one, **Gaussian Reweight**, is to emphasize pixels near the center, which is inspired by the Non-local Means [BCM05] in image processing.

We use a shallow convolutional neural network for feature embedding. The network architecture is shown in Figure 4.2. This network embeds $9 \times 9 \times 6$ neighborhood cubes into

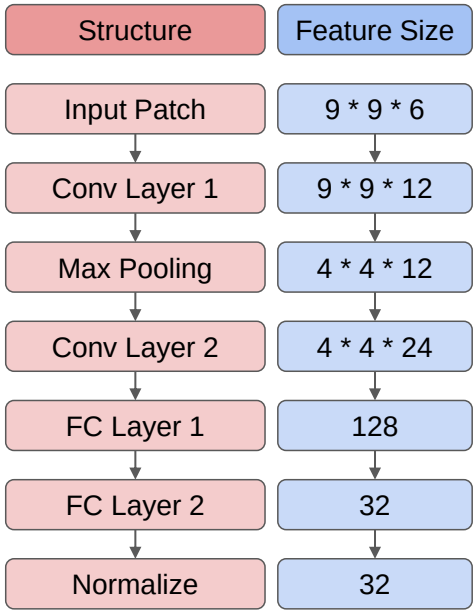


Figure 4.2: The architecture of our feature embedding neural network.

32-dimensional feature vectors. It only includes 57836 trainable parameters. **Conv Layer** denotes the convolutional layer and **FC layer** denotes the fully connected layer. There is a ReLU layer after each of **Conv Layer 1**, **Conv Layer 2**, and **FC Layer 1**. **Conv Layer 1** has the kernel size 5 and padding 2 while **Conv Layer 2** has the kernel size 3 and padding 1. The final layer normalizes the L2 norm of the output feature vector to one, which makes it easier for the angular similarity in the loss function (4.1), (4.2) of contrastive learning.

The loss function $\mathcal{L}(\theta)$ is the SupCon loss(4.2) for B-CGAP and is the SimCLR loss(4.1) for A-CGAP, since we don't have the ground-truth label information in the feature preprocessing stage of the A-CGAP. The training process is described in Section 4.2.1. Denote the trained parameters by $\hat{\theta}$. Finally, let the set of preprocessed feature vectors corresponding to image I in CGAP be written as $\{\mathbf{x}_i(I) = f(\mathbf{x}_i^{\text{raw}}(I), \hat{\theta}) \mid i = 1, 2, \ldots, N_0\}$.

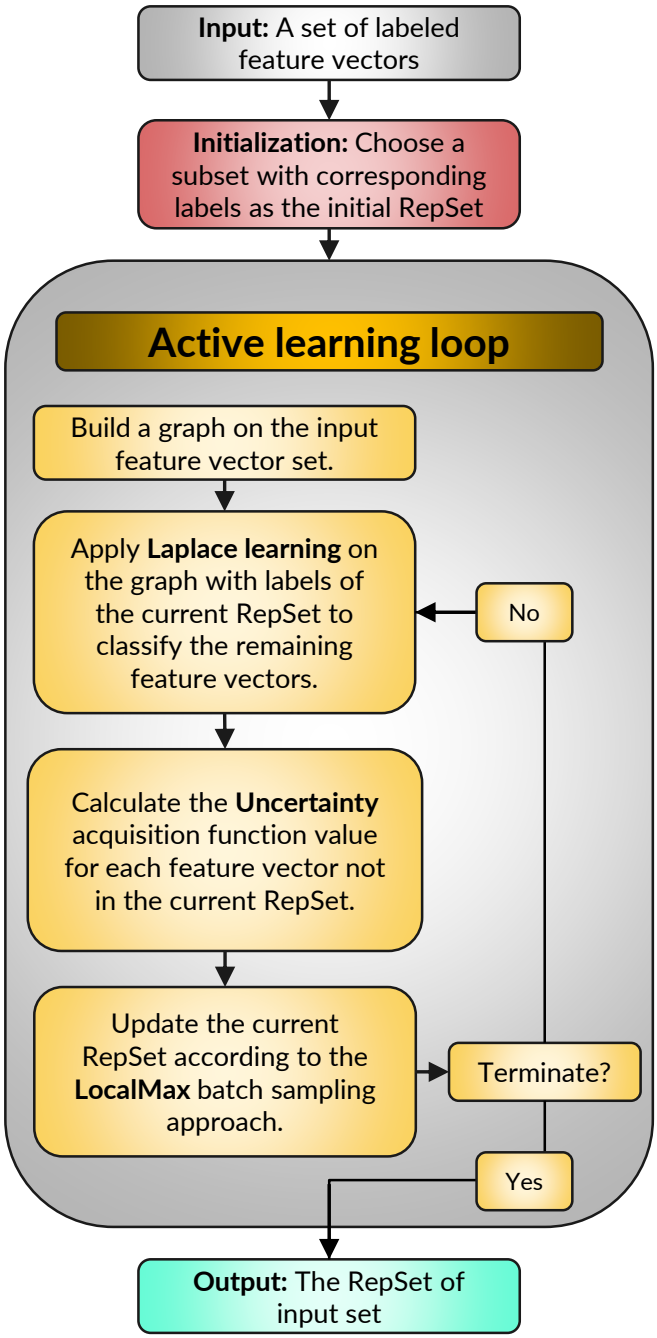


Figure 4.3: The creation of the RepSet with active learning. This process uses the graph Laplace learning [ZGL03], the Uncertainty acquisition function [BLS18, MLB20, QSW19], and the LocalMax batch active learning [CCT23] approaches.

4.2.3 Create the Representative Set

It is inefficient to include the massive set of all training feature vectors in graph learning. For example, considering label images of Kolyma, Yana, Waitaki, and Colville Rivers in the RiverPIXELS dataset, there are 42 multispectral images of the size $256 \times 256 \times 6$, consisting of over 2.7 million pixels. It would be prohibitively costly to generate a graph based on this extremely large number of feature vectors.

Denote X_i as the preprocessed feature vector set of the training image $I_i \in \mathcal{I}$. In this part, we condense the feature set $X = \cup_{i=1}^{n_l} X_i$ into a much smaller representative set (RepSet) $\mathcal{R} \subset X$ as a labeled training set that will be used to classify the pixels in each of the unlabeled images. Such a condensation process is expected to remove redundant feature vectors while keeping significant feature vectors.

We find a RepSet R_i for each image $I_i \in \mathcal{I}$ and subsequently union them together to obtain $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. In the case that ground-truth labels are not available when generating the RepSet $\mathcal{R}$ (i.e. the A-CGAP case), pixels need to be labeled by human experts in a **human-in-the-loop** process. To determine the RepSet for a certain image $I_i \in \mathcal{I}$, we follow 3 steps: **Initialization**, **Active learning loop**, and **Termination**. Such a process is illustrated in Figure 4.3.

We now present the details of these three steps:

1. Initialization: *Initialize the RepSet* R_i^0. There are two methods for initialization, **random initialization** and **core-set initialization**.

In the **random initialization**, feature vectors in the initial RepSet R_i^0 are randomly chosen from the set X_i. If all ground-truth labels of feature vectors in X_i are available, we can randomly sample the same number of feature vectors within each class to achieve a class-balanced initialization. If all ground-truth labels for the current image are *not available*, then the randomly sampled initial set will be highly imbalanced across labels. In many images in our dataset, land pixels account for over 80% of the image while sediment pixels account for

less than 5%. Random initialization likely yields little to no sediment pixels, exacerbating class imbalances in the subsequent **active learning loop**.

The **coreset initialization** select a core-set that follows the geometric distribution of the feature vector set X_0. Here we use the Dijkstra Annulus Core-Set (DAC), introduced in Section 3.2.1, which provides a good initialization for the graph-based active learning process. Compared with the **random initialization**, it is not efficient since it needs to construct a graph structure on X_i and down-sample according to this graph.

2. Active learning loop: *Following the guidance of active learning approaches (Section 2.3), add feature vectors from X_i and corresponding labels one by one to the RepSet.*

The **Active learning loop** is illustrated in Figure 4.3: Construct a graph G_i on X_i and initialize the RepSet to be R_i^0. Apply Laplace learning on G_i with the initially labeled set R^0 to make predictions for feature vectors in $X_i \setminus R^0$. Then based on the predicted labels, calculate the acquisition function $\mathcal{A}(\mathbf{x})$ for $\mathbf{x} \in X_i \setminus R^0$ according to the UC acquisition function (2.40). Then select a query set $\mathcal{Q}$ with the given batch size $|\mathcal{Q}| = B$ according to the LocalMax batch sampling approach introduced in 3.2.2. Update the RepSet as $R_i^1 = R_i^0 \cup \mathcal{Q}$. Repeat this process for each iteration t until reaching the terminal condition, to be explained in the next bullet entitled **Termination**. The final R_i^t is the RepSet of image I_i, denoted as R_i.

3. Termination: *Stop the active learning loop when a certain terminal condition is satisfied.* When ground-truth labels of all feature vectors are available, the **accuracy-based terminal condition** can be applied. Otherwise, the **label-change terminal condition** is applied.

In iteration t of the **active learning loop** step for image I_i, let X_i be the preprocessed feature set of pixels in I_i and R_i^t be the RepSet in the current iteration. We apply Laplace learning on the graph built with nodes X_i with labels on R_i^t to make predictions on $X_i \setminus R^t$. Let $\mathcal{Q}$ be the query set to obtain labels by active learning and $R_i^{t+1} = R_i^t \cup \mathcal{Q}$. The labels are either (1) already available if the training set images are fully labeled or (2) hand-labeled by

a human in the loop. Denote the Laplace learning prediction label on $\mathbf{x} \in X_i$ at iteration t by $y^t(\mathbf{x})$.

With a hyperparameter $K_{\max}, \epsilon$, these terminal conditions are based on the predicted labels on $X \setminus R_i^t, X \setminus R_i^{t+1}$ at iterations $t, t+1$.

With the δ-function 3.13, we present two kinds of terminal conditions to check if we need to terminate the process at iteration $t+1$:

1. **Accuracy-based terminal condition:** If the ground-truth label $y(\mathbf{x})$ is *available* for feature vector $\mathbf{x} \in X_i$, we can terminate the active learning loop according to the change in prediction accuracy. The accuracy at iteration t is calculated by:

$$a_t = \frac{\sum_{\mathbf{x} \in X_i \setminus R_i^t} \delta(y^t(\mathbf{x}), y(\mathbf{x}))}{|X_i \setminus R_i^t|}. \tag{4.3}$$

   Terminate the active learning loop if:

$$|a_t - a_{t+1}| < \epsilon \text{ or } t > K_{\max}. \tag{4.4}$$

   Practically, we further penalize low accuracy by applying a lower ϵ when a_{t+1} is relatively small. Given a fixed parameter γ, we use the terminal condition:

$$|a_t - a_{t+1}| < \epsilon \exp\left(-\frac{100(1-a_{t+1})}{\gamma}\right) \text{ or } t > K_{\max}. \tag{4.5}$$

2. **Label-change terminal condition:** If the ground-truth labels for feature vectors are *not available*, we can terminate the active learning loop according to the change of predicted labels. At iteration t, define the label-change value:

$$c_t = \frac{\sum_{\mathbf{x} \in X_i \setminus R_i^t} \delta(y^t(\mathbf{x}), y^{t-1}(\mathbf{x}))}{|X_i \setminus R_i^t|}. \tag{4.6}$$

   Terminate the active learning loop if:

$$c_{t+1} < \epsilon \text{ or } t > K_{\max}. \tag{4.7}$$

The accuracy-based terminal condition provides a clearer measure of how well a RepSet R_i^t is for the task of classifying the pixels in image I_i but requires the ground-truth labels for all pixels.

On the other hand, the label-change terminal condition does not require ground-truth labels so that one can create the RepSet from scratch. This terminal condition is met when the addition of labeled points to the current image's RepSet R_i^t changes the predicted labels of the unlabeled points $X \setminus R_i^t$ from Laplace learning at a very slow rate. In a sense, this measures when there is no significant marginal gain from adding more labeled points to the RepSet.

Such a process requires manually labeling only a few feature vectors to construct a RepSet. Such a process requires manually labeling only a few feature vectors to construct a RepSet. Remarkably, our experiments in the next section suggest that the size of the resulting RepSet is usually less than 1% of the original set of pixels in the training image set. Moreover, our pipeline consistently outperforms a range of fully-supervised methods in multiple experiments, even when operating with such a small labeled dataset.

4.2.4 Pipeline Structure

All our three pipelines, the GAP, B-CGAP, and A-CGAP, are trained on the training image set $\mathcal{I}$ and are used to classify pixels in the test image set $\tilde{\mathcal{I}}$. At the beginning of each pipeline, we preprocess each image in $\mathcal{I}$ and $\tilde{\mathcal{I}}$ according to section 4.2.2 into feature vectors. $X_i = \{x_1^i, x_2^i, \ldots, x_{N_0}^i\}$ and $\tilde{X}_j = \{\tilde{x}_1^j, \tilde{x}_2^j, \ldots, \tilde{x}_{N_0}^j\}$ denote the extracted feature set of image $I_i \in \mathcal{I}$ and $\tilde{I}_j \in \tilde{\mathcal{I}}$. Here $N_0 = 256^2 = 65536$ is the number of pixels in an image, which is common to each image in both training and test image sets. For the GAP, we use the non-local means feature vectors in $\mathbb{R}^{294}$ while we use the neural network embedding feature vectors in $\mathbb{R}^{32}$ for B-CGAP and A-CGAP.

The GAP and B-CGAP require all ground-truth labels in $\mathcal{I}$ in the training process. Fig-

ure 4.1 presents the flowchart of the B-CGAP. For B-CGAP, we train the feature embedding neural network with the **supervised contrastive loss** (4.2). **The following steps are the same for GAP and B-CGAP; the only difference between them is the feature preprocessing.** For each $I_i \in \mathcal{I}$, since all ground-truth labels of feature vectors in its feature set X_i are given, the RepSet R_i can be condensed from X_i with the class-balanced **random initialization** and the **accuracy-based terminal condition** according to Section 4.2.3. The RepSet of $\mathcal{I}$ is $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. For each test image $\tilde{I}_j \in \tilde{\mathcal{I}}$ with feature set $\tilde{X}_j$, construct a graph $G_j = (\mathcal{R} \cup \tilde{X}_j, W_j)$, where $\mathcal{R} \cup \tilde{X}_j$ is the vertex set of size $N_j = |\mathcal{R} \cup \tilde{X}_j|$ and W_j is the weight matrix generated from feature vectors in $\mathcal{R} \cup \tilde{X}_j$ according to Section (2.2.1). Apply Laplace learning on G_j to get the classifier matrix $U_j \in \mathbb{R}^{N_j \times n_c}$ according to Section 2.2.2. The predicted label $\tilde{y}_k^j$ of each unlabeled feature vector $\tilde{x}_k^j \in \tilde{X}_j$ is given by

$$\tilde{y}_k^j = \arg\max \mathbf{u}_j(\tilde{x}_k^j), \ k = 1, 2, \ldots, N_0, \tag{4.8}$$

where the arg max of a vector $\mathbf{u}$ is the (first) index of $\mathbf{u}$'s largest element.

In addition, we include an extension of the B-CGAP, called the adaptive GAP (A-CGAP). The training process of the A-CGAP does *not* require a priori access to the ground-truth labels for all pixels in each training image $I_i \in \mathcal{I}$. We apply the **SimCLR contrastive loss** (4.1) to train the feature embedding network. The creation of RepSet requires the **coreset initialization** and the **label-change terminal condition** according to Section 4.2.3 to sample RepSets R_i for $I_i \in \mathcal{I}$. With the aid of a human in the loop during this process, feature vectors in R_i are manually labeled in the active learning process. Let $\mathcal{R} = \cup_{i=1}^{n_l} R_i$. *The rest steps of the A-GAP are the same as the B-GAP.* We construct a graph on $\mathcal{R} \cup \tilde{X}_j$ and use graph Laplace learning to classify feature vectors in $\tilde{X}_j$.

It should be noticed that the A-CGAP provides flexibility for applications wherein there is no predefined training image set $\mathcal{I}$, but rather just a test image set $\tilde{\mathcal{I}}$. In such a case, the A-CGAP extracts a RepSet $\tilde{\mathcal{R}}$ of $\tilde{\mathcal{I}}$ from scratch and applies Laplace learning with the human-in-the-loop labels for $\tilde{\mathcal{R}}$ to classify the other pixels in $\tilde{\mathcal{I}}$. Furthermore, A-CGAP could be applied to reinforce the B-CGAP. Suppose one has access to all ground-truth labels for

pixels in $\mathcal{I}$. In that case, we can apply the B-CGAP, and the A-CGAP to respectively extract the RepSets $\mathcal{R}$ and $\tilde{\mathcal{R}}$ of $\mathcal{I}$ and $\tilde{\mathcal{I}}$. Let $\mathcal{R}_{\text{new}} = \mathcal{R} \cup \tilde{\mathcal{R}}$ be the new RepSet. For an image $\tilde{I}_j \in \tilde{\mathcal{I}}$, applying Laplace learning on the graph with vertices $\mathcal{R}_{\text{new}} \cup \tilde{X}$ with labeled set $\mathcal{R}_{\text{new}}$ to classify feature vectors in $\tilde{X} \setminus \mathcal{R}_{\text{new}}$. Such a process allows us to expend limited human-in-the-loop effort to expand our labeled feature set (RepSet) by including some feature vectors from the test image set.

4.3 Experiments and Results

This section includes experiments of our pipelines on the RiverPIXELS dataset. We choose five rivers from the dataset: the Kolyma, Yana, Waitaki, Colville, and Ucayali Rivers. Our pipelines are trained on images chosen from the first four rivers while the performance of different methods is tested on all these five rivers. There are 42 images belonging to the first four river regions, and the Ucayali River includes 54 images. For each region, we randomly sample 75% of the labeled data as the training set and use the remaining 25% as the test set. The training set $\mathcal{I}$ has 32 labeled images, while the test set $\tilde{\mathcal{I}}$ has 10 labeled images, which are considered unlabeled in our experiments. In Section 4.3.2, the test set image set $\tilde{\mathcal{I}}_{\text{ex}}$ is formed by 54 images of the Ucayali river.

Various metrics are provided to evaluate the performance of different methods and schemes. Each unlabeled image $\tilde{I}_j \in \tilde{\mathcal{I}}$ has M^2 pixels $\{p^j_{k,l}\}_{k,l=1}^M$ with ground-truth labels $\{y^j_{k,l}\}_{k,l=1}^M$ and predicted labels $\{\bar{y}^j_{k,l}\}_{k,l=1}^M$, where $M = 256$. We define d_b to be the distance to the boundary for each pixel $p^j_{k,l}$ of coordinate k, l of image $\tilde{I}_j$ by

$$d_b(p^j_{k,l}) = \min_{\{(\hat{k},\hat{l}):y^j_{k,l} \neq y^j_{\hat{k},\hat{l}}\}} \sqrt{(k-\hat{k})^2 + (l-\hat{l})^2}, \tag{4.9}$$

which is the Euclidean distance to the nearest pixel with a different ground-truth label.

For the test set $\tilde{\mathcal{I}}$ (or $\tilde{\mathcal{I}}_{\text{ex}}$) with the size $|\tilde{\mathcal{I}}| = n_u$, we define following metrics:

1. **Overall Accuracy:** The overall accuracy is the average accuracy of all pixels.

$$\text{Overall} = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, \bar{y}_{k,l}^j)}{n_u M^2} \tag{4.10}$$

2. **Class Accuracies:** We consider the true positive rate (TPR), false positive rate (FPR), and the normalized false positive rate (NFPR) of each class. For class index c:

$$\text{TPR}(c) = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, c)\delta(\bar{y}_{k,l}^j, c)}{\sum_{j=1}^{n_t} \sum_{k,l=1}^{M} \delta(y_{k,l}^j, c)} \tag{4.11}$$

$$\text{FPR}(c) = \frac{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} \delta(\bar{y}_{k,l}^j, c)(1 - \delta(y_{k,l}^j, c))}{\sum_{j=1}^{n_u} \sum_{k,l=1}^{M} (1 - \delta(y_{k,l}^j, c))}. \tag{4.12}$$

3. **Boundary Accuracy** The boundary accuracy of distance d is the average accuracy of pixels whose distance to the boundary is less or equal to d.

$$\text{BA}(d) = \frac{\sum_{j=1}^{n_u} \sum_{\{(k,l):d_b(p_{k,l})\leq d\}} \delta(y_{k,l}^j, \bar{y}_{k,l}^j)}{\sum_{j=1}^{n_u} |\{(k,l) : d_b(p_{k,l}) \leq d\}|} \tag{4.13}$$

Remark 11. *While we do report* ***Overall Accuracy*** *and* ***Class Accuracies****, the* ***Boundary Accuracy*** *metrics may provide more meaningful insight into the models' performance. The land, water, and sediment classes in both our selected images and in general are imbalanced, with land pixels accounting for 70% to 90% of each image. Therefore, a naive classifier that tends to simply classify pixels primarily as land may still report an excellent* ***Overall Accuracy****. Furthermore, as a method, we will rarely have both the best TPR and FPR for every single class – i.e., the best performance in each of the* ***Class Accuracies*** *metrics. We suggest that the* ***Boundary Accuracy*** *metric is the most indicative of model performance.*

In this section, we compare our proposed GAP, B-CGAP and A-CGAP to various other methods, such as DeepWaterMap (DWM) [IBP19], support vector machine (SVM) [CV95] and random forest (RF) [Ho95]. After feature preprocessing (Section 4.2.2), the original extracted feature vector set X has over 2 million feature vectors from the training set consisting

of 32 labeled images. Each method has a different amount of training data–the training data is chosen to represent the best performance of each method. The training and test datasets information for each method are listed in Table 4.1. This table applies to all experiments throughout Section 4.3 if not specifically mentioned. We now present the **training set details** for each of the considered methods:

For SVM and RF, the training performances are almost the same when the number of training feature vectors is relatively large. To balance the training performance and the computational cost of model fitting, we randomly select a subset of all labeled pixels to train SVM and RF. For each labeled image in the training set $\mathcal{I}$, we randomly sample N_s pixels from each class (if a class has less than 500 pixels, sample all) to form a pixel set $\mathcal{P}$. The pixel set $\mathcal{P}$ includes 42634 pixels consisting of 16000, 14558, and 12076 pixels for land, water, and sediment respectively.

For our GAP, B-CGAP, or A-CGAP methods, the training set for the graph Laplace learning is the representative set (RepSet) $\mathcal{R}$ extracted from X through the LocalMax batch active learning process with the batch size $B = 15$. For B-CGAP and GAP, the **accuracy terminal condition**(4.3)(4.5) is applied with the $\epsilon = 10^{-4}$, $K_{\max} = 3000$ (for each training image), and $\gamma = 5$ (for B-CGAP only). For A-CGAP, the **label-change terminal condition**(4.6)(4.7) is applied with $\epsilon = 5 \times 10^{-4}$, $K_{\max} = 3000$ (for each training image).

DeepWaterMap [IBP19] only provides the classification of water and land pixels, while RiverPIXELS patches include water, bare sediment, and land. Here the "land" does not follow a strict definition, which can include complicated ground textures like buildings, mountains, and forests. In binary classification results, such as DWM, "land" refers to non-water pixels. In 3-class classification results including the sediment, such as our pipeline, "land" refers to non-water and non-sediment pixels. As a result, we cannot directly compare our three-class model with DWM. Here, we provide two approaches to compare the performance of the other methods and DWM.

The first approach is to *retrain DeepWaterMap* (DWM-R). We train a new neural network

with the same structure of DWM on our training set with 32 labeled images and labels of water, sediment, and land. The resulting CNN thus provides a classification of water, sediment, and land pixels. The second approach is to *modify labels*. Inspection of the original DeepWaterMap (DWM-O) training set shows that nearly all sediment pixels are labeled as land. We modify the labels of our training set and the ground-truth labels of our test set by changing sediment labels to land labels. Based on this modification, we train all methods as classifiers for water and land and compare them with the original DeepWaterMap.

In light of these details regarding DWM, we consider two types of training sets for DWM comparisons. DWM-R is trained on the training image set $\mathcal{I}$ that all of our other comparison methods are trained on. DWM-O is trained on the vast training set of the original DeepWaterMap [5].

It should be noted further that the pixel-wise feature vectors for different methods might differ. B-CGAP uses the supervised contrastive learning (SupCon (4.2)) neural network embedding feature vectors while A-CGAP uses the unsupervised (SimCLR (4.1)), according to Section 4.2.1, 4.2.2. We also provide experiments with SVM and RF results, marked by the suffix "-E", on the SupCon feature vectors. The GAP, SVM, and RF are based on the 294-dimensional Non-local means feature vectors [BCM05, CMB23b] generated by 7×7 neighborhood patches centering at each pixel. For DWM, the inputs are images rather than feature vectors since DWM is a CNN-based method that takes the whole image as input.

4.3.1 Comparison between different methods

We compare the classification performance of our GAP, B-CGAP, and A-CGAP to DWM [IBP19], SVM [CV95], and RF [Ho95] models. Information regarding training and test sets is shown in Table 4.1. For this subsection (Subsection 4.3.1), the test set is denoted as $\tilde{\mathcal{I}}$ and contains 10 images. It is worth noting that our methods, B-CGAP, A-CGAP, and GAP,

[5] `https://github.com/isikdogan/deepwatermap`

Information on Training and Test Datasets

Method	***Original Set***		***Network***	***Sampled Training Set***		***Test Set***	
	Dataset	*Num Images*	***Embedding***	*Dataset*	*Num Pixels*	*Dataset*	*Num Images*
B-CGAP (Ours)	$\mathcal{I}$	32	SupCon	RepSet	3.71K	$\tilde{\mathcal{I}}$ for Section 4.3.1	$\tilde{\mathcal{I}} = 10$
A-CGAP (Ours)			SimCLR	RepSet	2.99K		
GAP (Ours)			No	RepSet	3.27K		
SVM [CV95]			No	$\mathcal{P}$	42.6K	$\tilde{\mathcal{I}}_{\text{ex}}$ for Section 4.3.2	–
SVM-E [CV95]			SupCon	$\mathcal{P}$	42.6K		$\tilde{\mathcal{I}}_{\text{ex}} = 54$
RF [Ho95]			No	$\mathcal{P}$	42.6K		
RF-E [Ho95]			SupCon	$\mathcal{P}$	42.6K	No test set for Sections 4.3.3, 4.3.4	
DWM-R [IBP19]			No	$\mathcal{I}$	2.1M		
DWM-O [IBP19]	$\mathcal{I}_{\text{DWM}}$	100K	No	$\mathcal{I}_{\text{DWM}}$	6.55B		

Table 4.1: Information of training and test datasets of different methods implemented in the experiments in this chapter. Methods implemented here are B-CGAP, A-CGAP, GAP, SVM, RF, retained DWM (DWM-R), and original DWM (DWM-O). SVM and RF's suffix "-E" corresponds to using the neural network embedding features. The training set of DWM-O is the original training set of DeepWaterMap, while other methods use training sets sampled from RiverPIXELS. The test sets used in different sections of this chapter vary, with $\tilde{\mathcal{I}}$ used in Section 4.3.1, $\tilde{\mathcal{I}}_{\text{ex}}$ used in Section 4.3.2, and no test set used in Sections 4.3.3 and 4.3.4. K, M, and B denote thousands, millions, and billions, respectively.

Comparison Approach: Retrain DeepWaterMap

All values in Percentage (%)

Importance	***3rd***						***1st***		***2nd***
Method	***Land***		***Water***		***Sediment***		***Boundary***		***Overall***
	TPR	*FPR*	*TPR*	*FPR*	*TPR*	*FPR*	*BA(3)*	*BA(10)*	*OA*
B-CGAP (ours)	98.57	5.63	92.29	1.59	78.49	**0.75**	**88.85**	**92.29**	**96.61**
A-CGAP (ours)	**99.02**	11.25	86.93	**1.36**	64.28	**0.77**	84.51	90.35	95.35
GAP (ours)	98.27	8.43	89.89	1.98	65.72	0.86	81.90	90.41	95.50
SVM [CV95]	95.09	6.20	89.74	4.10	82.89	1.75	79.28	88.05	93.55
SVM-E [CV95]	91.78	**0.63**	93.84	6.61	**96.70**	2.42	78.52	85.26	92.38
RF [Ho95]	93.72	2.65	92.26	5.10	90.73	2.14	77.40	86.84	93.31
RF-E [Ho95]	91.81	0.95	**94.27**	6.94	92.49	2.06	80.23	86.32	92.38
DWM-R [IBP19]	97.38	14.53	83.99	3.12	41.70	1.08	72.56	84.56	92.86

Table 4.2: The comparison among different methods trained as 3-class classifiers of the land, water, and sediment. This table compares our B-CGAP, A-CGAP, GAP, SVM, RF, and retrained DWM (DWM-R). SVM and RF include both the non-local means feature vectors and the neural network embedding feature vectors (-E). Accuracy metrics include the true positive rate (TPR), false positive rate (FPR) of each class, the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy (OA). The first row "importance" indicates the important ranking of three different types of accuracy metrics. The best one of each accuracy metric (each column) is bolded. Our B-CGAP performs the best on boundary accuracies and the overall accuracy.

Comparison Approach: Modify Labels (Sed → Land)

All Values in Percentage (%)

Importance	***3rd***						***1st***		***2nd***
Method	***Land***		***Water***		***Sediment***		***Boundary***		***Overall***
	TPR	*FPR*	*TPR*	*FPR*	*TPR*	*FPR*	*BA(3)*	*BA(10)*	*OA*
B-CGAP(ours)	98.58	8.27	91.73	1.42	N/A	N/A	**89.73**	**93.66**	**97.03**
A-CGAP(ours)	**99.04**	14.80	85.20	**0.96**	N/A	N/A	85.46	91.17	95.90
GAP (ours)	98.58	11.47	88.53	1.42	N/A	N/A	83.75	91.66	96.30
SVM [CV95]	97.73	12.49	87.51	2.27	N/A	N/A	82.77	91.08	95.41
SVM-E [CV95]	94.09	6.10	93.90	5.91	N/A	N/A	81.67	87.80	94.04
RF [Ho95]	96.37	9.37	90.63	3.63	N/A	N/A	81.50	89.63	95.07
RF-E [Ho95]	93.44	**5.93**	**94.07**	6.56	N/A	N/A	82.67	88.18	93.58
DWM-O [IBP19]	97.85	14.26	85.74	2.15	N/A	N/A	78.02	88.81	95.11

Table 4.3: The comparison among different methods trained as 2-class classifiers of the land and water. To compare with the original DeepWaterMap (DWM-O), we changed all ground-truth labels of sediment into land. This table compares our B-CGAP, A-CGAP, GAP, SVM, RF, and original DWM (DWM-O). Accuracy metrics include the true positive rate (TPR) and false positive rate (FPR) of each class, the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy (OA). The first row "importance" indicates the important ranking of three different types of accuracy metrics. The best one of each accuracy metric (each column) is bolded. Our B-CGAP performs the best on boundary accuracies and overall accuracy. *It is a coincidence that the B-CGAP and GAP have the same land TPR and Water FPR.*

use **significantly less** training data than the other methods considered. The results are presented in two tables: Table 4.2 shows the comparison to the retrained DWM (DWM-R) as a 3-class classifier, while Table 4.3 compares to the original DWM (DWM-O) when the sediment class in the RiverPIXELS dataset is modified to be classified as land.

Figures 4.4, 4.5, and 4.6 are sampled images and experiment results of the Waitaki, Colville, and Yana Rivers, respectively.

In summary, according to Tables 4.2, 4.3, our approach B-CGAP has the best performance measured by BA(3), BA(10), and OA. Our GAP and A-CGAP perform similarly and have the second good result. According to Table 4.1, compared with other methods, all three methods, B-CGAP, A-CGAP, and GAP, are trained on a much smaller training set, which only takes around 0.15% pixels of the training set $\mathcal{I}$. Although A-CGAP performs similarly to our previous pipeline, GAP, A-CGAP does not require any ground-truth information at the beginning of the training process.

Here we provide an analysis of the overall strengths of the methods.

Discussion 1 – Compare with DWM

In Table 4.2, the retrained DWM (DWM-R) has the worst performance in both BA(3) and BA(10). This implies that our training set $\mathcal{I}$ with 32 images is not sufficient to train a good CNN-UNet. In Table 4.3, the original DWM (DWM-O) has a similar performance to the SVM and RF. However, it can not provide the prediction of the sediment class.

Discussion 2 – the neural network embedding

The main difference between our B-CGAP and the previous GAP is the network embedding. According to Tables 4.2, 4.3, there is a big improvement from GAP to B-CGAP. However, the SVM-E and RF-E perform worse than SVM and RF, respectively, based on the same SupCon embedding used by B-CGAP. There are significant increases in the water and sediment TPR and FPR when using the SupCon network embedding feature vectors. A possible reason for this observation is that the graph learning classifier based on the angular

similarity aligns better with the similarity metric in the contrastive loss functions (4.1), (4.2).

Discussion 3 – B-CGAP v.s. A-CGAP Our A-CGAP does not require any ground-truth information at the beginning of the training process (Section 4.2.4). One major difference is the training process of the feature embedding neural network – A-CGAP uses the self-supervised loss SimCLR (4.1) while B-CGAP uses the supervised loss SupCon(4.2). In A-CGAP, only the 2.99K training feature vectors in the RepSet require ground-truth labeling, which can be processed by a human-in-the-loop process.

4.3.2 Performance on other regions

In the previous Section 4.3.1, all methods are trained on subsets of $\mathcal{I}$, and the performances are evaluated on the test set $\tilde{\mathcal{I}}$. We note that $\mathcal{I}$ and $\tilde{\mathcal{I}}$ are segmented from the same set, the Kolyma, Yana, Waitaki, and Colville rivers. For a certain image $\tilde{I}_j \in \tilde{\mathcal{I}}$, there is another image in $I_i \in \mathcal{I}$ that belongs to the same region as $\tilde{I}_j$, where the region refers to either the Arctic or New Zealand. Since DeepWaterMap works globally, we want to test if our models also retain accuracy in other regions that may have different landscapes or geographic features.

We consider images of the Ucayali River in our RiverPIXELS dataset. The Ucayali River is a tributary to the Amazon River and is mostly single-threaded with large in-channel bare sediment bars. We apply the same comparison strategies as in section 4.3.1. The training information of each method is the same as that in Section 4.3.1, Table 4.1, while the test set now is the set of images of the Ucayali river, which includes 54 images. Table 4.4 compares our B-CGAP to GAP, SVM, RF, the retrained DWM (DWM-R) and original DWM (DWM-O) on the test set $\tilde{\mathcal{I}}_{\text{ex}}$. Similarly to Section 4.3.1, we provided results on 3 classes of land, water, and sediment, and 2 classes of water and non-water by modifying the sediment labels into the land to compare with both the DWM-R and DWM-O. According to Table 4.4, our B-CGAP has the best performance as we have the highest BA(3), BA(10), and overall accuracy in both comparisons.

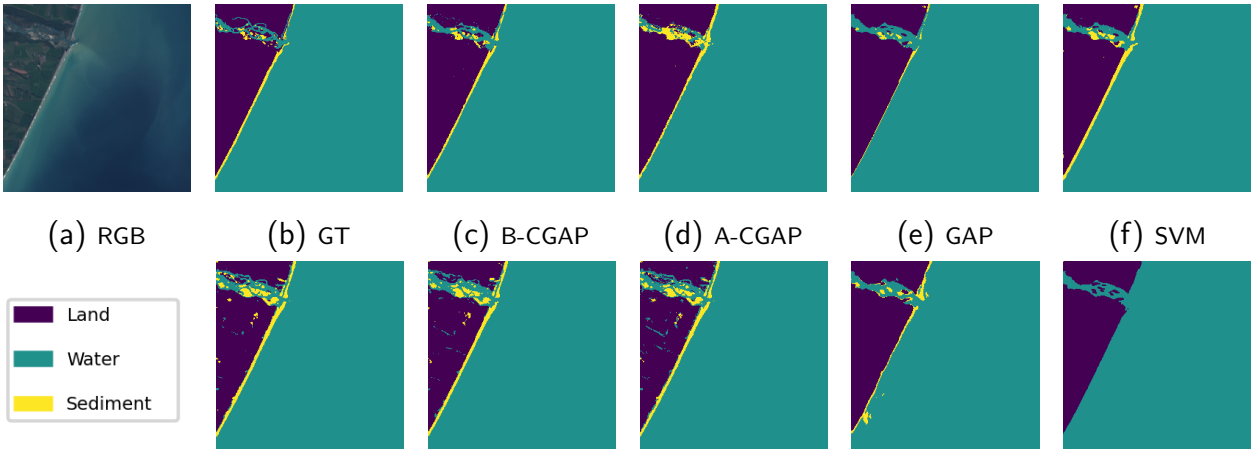


Figure 4.4: Results for a Patch of Waitaki River. Original Patch name: "Waitaki_River_1 2019-03-02 074 091 L8 413 landsat". This Patch contains an estuary and a coastline. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.

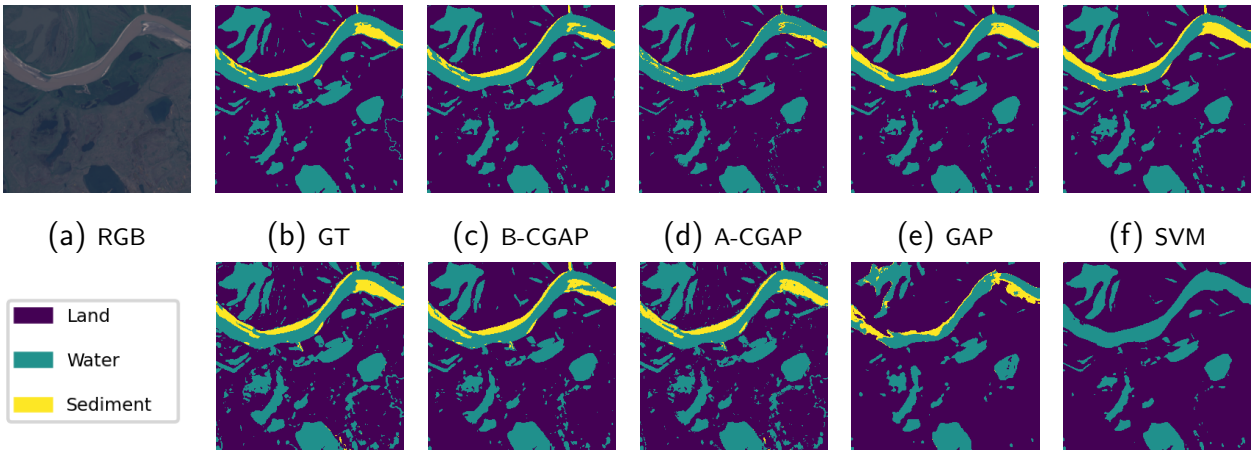
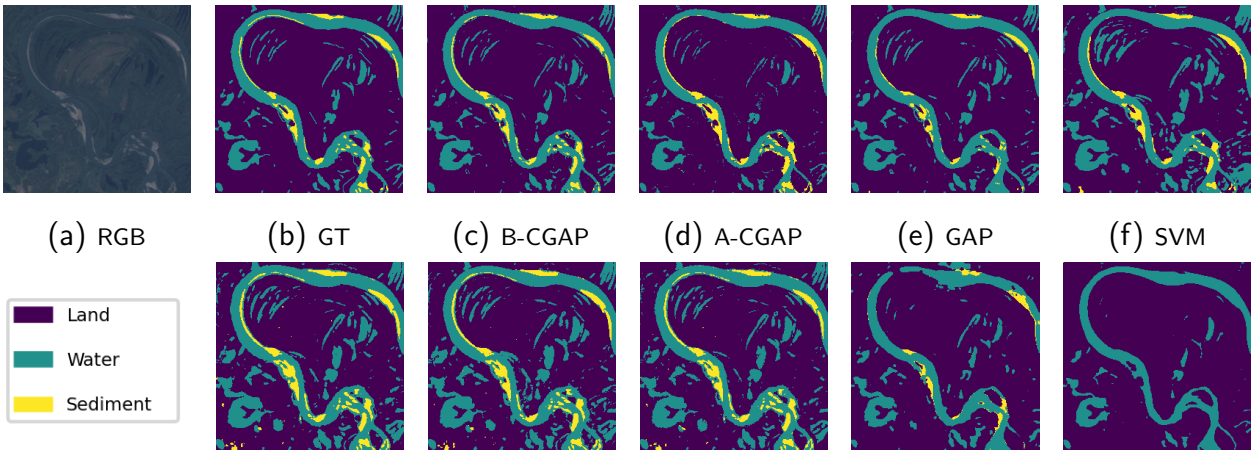


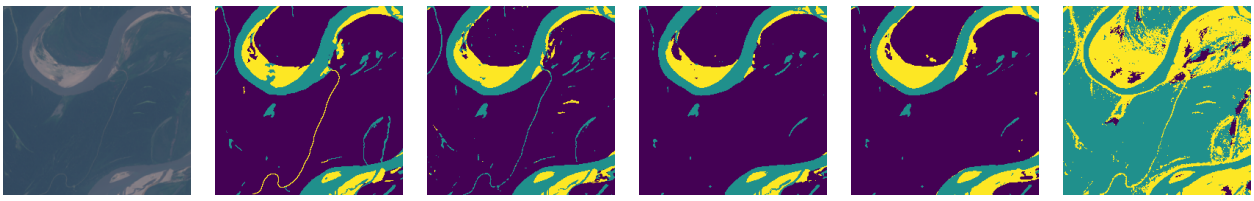
Figure 4.5: Results for a Patch of the Colville River. Original Patch name: "Colville_River_2 2015-07-11 076 011 L8 125 landsat". This Patch contains a complex network of water, including a mainstream, some lakes and small tributaries. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.



(a) RGB (b) GT (c) B-CGAP (d) A-CGAP (e) GAP (f) SVM

(g) SVM-E (h) RF (i) RF-E (j) DWM-R (k) DWM-O

Figure 4.6: Experiment on images of Yana river. Original "Yana_River_1 1991-08-13 122 012 L5 511 landsat". This Patch contains a complex network of water, including a mainstream, some lakes and small tributaries. Panel (k) is the original DWM prediction for water and non-water pixels while other panels (b)-(j) are 3-class results of land, water, and sediment.



(a) RGB (b) GT (c) B-CGAP (d) GAP (e) SVM (f) RF

Figure 4.7: Results for the Ucayali River. Original Patch name: "Ucayali_River_1 2018-09-11 006 066 L8 549 landsat". This Patch includes two mainstreams and some small tributaries. Purple, cyan, and yellow represent land, water, and sediment respectively.

Experiments on Images of the Ucayali river

Method	***Retrain DWM***(%)			***Sed→Land***(%)		
	BA(3)	*BA(10)*	*OA*	*BA(3)*	*BA(10)*	*OA*
B-CGAP (ours)	**90.63**	**95.29**	**98.31**	**91.57**	**95.40**	**98.60**
GAP (ours)	83.07	92.93	97.48	85.08	94.17	98.21
SVM [CV95]	79.08	90.29	96.26	77.49	90.96	97.23
RF [Ho95]	26.41	26.79	12.52	76.01	89.09	96.21
DWM-R [IBP19]	69.28	83.43	94.39	N/A	N/A	N/A
DWM-O [IBP19]	N/A	N/A	N/A	79.42	91.14	97.29

Table 4.4: This table shows the comparison of our GAP and B-CGAP to SVM, RF, DWM-R, and DWM-O. Accuracy values in this table are based on the extra test set $\tilde{\mathcal{I}}_{\text{ex}}$ consisting of 54 images of the Ucayali river, while methods in this table are trained on the training set $\mathcal{I}$ of the Arctic and New Zealand. More details of the training set refer to Table 4.1. Metrics are the boundary accuracy of distances 3 and 10 (BA(3), BA(10)), and the overall accuracy. The best one of each accuracy metric (each column) is bolded. To compare with the retrained DWM (DWM-R) and the original DWM (DWM-O), we provide results on 3 classes (columns "Retrain DWM") and 2 classes (columns Sed→Land). Our B-CGAP performs the best on boundary accuracies and overall accuracy.

We sample two images of the Ucayali river and show those results in figures 4.7 and 4.8. According to these figures and tables, the random forest method (RF) completely fails, implying that it may be unstable when applied to a region different from its training set. In Figure 4.7, our B-CGAP detects the small tributaries better than other methods. In Figure 4.8 with light cloud haze, both our B-CGAP and GAP method (panel (c), (d)) well-match the ground-truth labels (panel (b)).

4.3.3 Efficiency Analysis

From previous comparisons among different methods, we conclude that our B-CGAP has the best performance, and A-CGAP and GAP perform the second. In this part, we provide information on the time consumption of three methods, B-CGAP, A-CGAP, and GAP, in both training and deploying. We choose only these three methods because they have a similar pipeline of graph-based active learning and they perform better than other methods.

Table 4.5 shows the time consumption information. It can be seen that with the low-dimensional feature vectors preprocessed by the neural network (our B-CGAP and A-CGAP), the active learning and model deploying processes are significantly accelerated. The model deploying becomes **more than 10 times faster** compared with our GAP method.

Here, we train both the B-CGAP and A-CGAP preprocessing networks on the whole RiverPIXELS dataset of 104 images to make a fair comparison. In previous Sections 4.3.1 and 4.3.2, the B-CGAP is trained only on the training set $\mathcal{I}$ of 32 images since it is supervised. Since the whole dataset includes 6.8M pixels and the dataset $\mathcal{I}$ includes 2.1M pixels, we use part of them for the network training. Practically, we use all the sediment pixels, randomly sampled 20% water pixels, and randomly sampled 3% land pixels to train the neural network, which is a subset of 0.48M pixels.

As for the hyperparameters, we choose the constant parameter $\tau = 0.5$ in the loss functions (4.1),(4.2), learning rate 0.02, and batch size 2048 to train both networks for 200 epochs.

4.3.4 Low-dimensional Visualization of Feature Vectors

Here we would like to provide more details on how the feature embedding neural network changes the feature vectors. Figure 4.9 shows the low-dimensional visualization of different feature vectors using UMAP [MHS18] and t-SNE [MH08] methods.

In Figure 4.9, panels (a) and (d) are visualizations of raw neighborhood patches of pixels.

Time Consumption

Method	***Info***		***Training***				***Deploying***		
	CPU	*GPU*	***Feature NN***		***Active Learning***		*Image Size*	*NN*	*Graph Learning*
			Epochs	*Time*	*Batch*	*Time*			
B-CGAP	i7-11800H	RTX 3070	200	14.34h	15	2023.21s	256×256	0.448s	10.76s
A-CGAP)			200	13.87h		1710.55s		0.448s	11.15s
GAP			N/A	N/A		6881.73s		N/A	135.31s

Table 4.5: This table shows the time consumption for our GAP, B-CGAP, and A-CGAP. The neural network (NN) training and deploying stages use the GPU, and all other processes are on the CPU. Although the neural network training takes a relatively long time, it reduces both the active learning time and model deploying time significantly.

We can discern approximately three cluster structures in (a) and (d), but there is a clear mix and overlap at the center and between the boundaries of each pair of classes. For the embeddings produced by SimCLR (panels (b), (e)), the cluster structures are more defined, with clearer distinctions between boundaries. However, the clusters are a bit dispersed, for instance, the water class (cyan) is divided into three smaller clusters in both visualizations. The low-dimensional visualization by the SupCon method is the best, with the clearest boundaries between classes and each class generally forming a cohesive cluster.

4.4 GraphRiverClassifier: A Global Classifier for Water and Sediment Pixels in Satellite Images

We provide a Python-based demo, called GraphRiverClassifier (GRC), to classify any Landsat-5 image (GitHub repository[6]). Details of how to use the demo refer to the `readme.md` file in the repository. In this section, we describe some significant features of this tool.

[6] `https://github.com/wispcarey/GraphRiverClassifier`

4.4.1 Google Platforms

The tool described is implemented in Python and utilizes Google Earth Engine (GEE)[7] for identifying and downloading the requested Landsat scenes. Users can perform the image extraction by simply providing the coordinates of the center point along with the desired latitude and longitude range of the bounding box. The selection method for Landsat scenes (e.g. surface reflectance versus top of atmosphere) is fully consistent with the RiverPIXELS dataset [SR22], to which their readme can be referred for further information.

Once Landsat scenes are identified, the tool automatically employs a pre-trained feature embedding neural network to preprocess feature vectors. Subsequently, our B-CGAP method is deployed to classify the pixels within the image into land, water, and sediment. The neural network training and the selection of RepSet through graph-based active learning are based on the RiverPIXELS dataset.

It is recommended that the tool be run on Google Colab as it allows for a seamless connection with GEE and ensures the automation of the entire process. Running the tool via a Google Colab notebook also avoids large image downloads by reading directly from a GEE-connected Google Drive account. Users may run the tool locally, but Landsat scene identification and downloading must be done manually. The tool also offers an integration with ChatGPT that allows users to enter the name of a place or river to return a bounding box. However, this feature requires access to ChatGPT's API and may incur costs.

4.4.2 Global Classifier of Water and Sediment

The GRC tool offers advanced classification solutions for water and sediment across the globe, leveraging the power of remote sensing and machine learning. This tool stands out for its exceptional flexibility, ease of use, and global scope, allowing users not only to define custom geographic areas but also to choose from a variety of Landsat datasets and temporal

[7] `https://earthengine.google.com/`

ranges. Such versatility ensures that users can conduct detailed and specific analyses tailored to their research or management needs. We note that the workflow of the tool also enables rapid development and testing of new classification algorithms.

In terms of processing speed, our GRC demonstrates impressive efficiency. For an image with one million pixels, the full process typically takes between 3 to 5 minutes. This includes data identification and download from GEE and storage to your Google Drive, which takes approximately 60 to 90 seconds, followed by feature embedding with a neural network and classification via graph learning, taking around 180 seconds in total. Figures 4.10, 4.11 show the visualization of two regions about the Ucayali and Murray Rivers that do not belong to the RiverPIXELS dataset.

4.4.3 Robustness to Different Resolutions

According to Section 4.2.2, the augmentations we designed for contrastive learning allow the embedding feature vectors to be robust with different resolutions. The parameter "scale" allows us to extract images of different resolutions on the same region. In Figures 4.10, 4.11, we present results of three resolutions, 15-meter, 30-meter, and 60-meter. The classification results are almost the same in panels (b)-(d), which verifies the robustness of our feature preprocessing approach.

4.5 Conclusion and Future Directions

We develop graph-based active learning pipelines to detect surface water and sediment pixels in multispectral images. The GAP method applies the non-local mean approach to extract pixel-wise feature vectors. We extend GAP to CGAP by employing a contrastive learning approach to train a shallow feature embedding neural network to preprocess raw neighborhood patches of each feature. The contrastive learning method is compatible with both supervised and unsupervised scenarios using SupCon or SimCLR loss functions. It offers two

significant advantages: first, based on our custom augmentations, the processed embedding feature vectors become more robust to different resolutions, cloud coverage, and geometric transformations; second, such a neural network significantly reduces the dimensionality of the feature vectors, greatly improving the efficiency of subsequent steps.

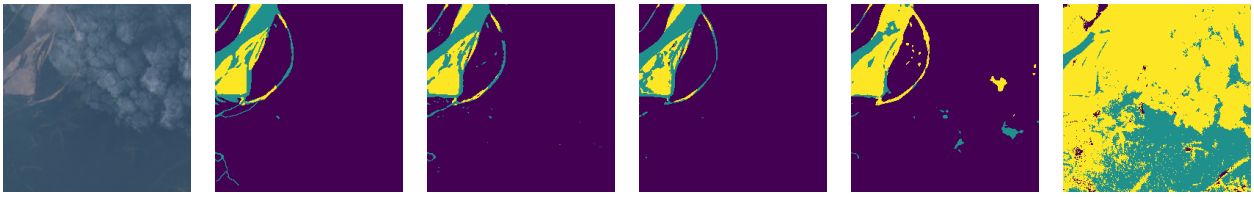
Our methods' most significant feature is their minimal data requirement, using only 3,000 training vectors to surpass the performance of CNN neural networks trained with 2.1 million pixels and outperforming SVM and RF models trained with larger datasets. Through our experiments, the CGAP method has significantly improved efficiency and accuracy compared to GAP. Furthermore, we introduced two versions, B-CGAP and A-CGAP, where the former is suitable for situations with complete ground-truth information, while the latter is useful for training from scratch with no ground-truth information. A-CGAP identifies the pixels most crucial to the graph-learning model and asks a human-in-the-loop to provide these labels. Experiments show that A-CGAP's performance is similar to our previously proposed GAP method, slightly inferior to B-CGAP.

We provide a Python-based tool, GraphRiverClassifier (GRC), to detect surface water and sediment globally. This tool utilizes Google Earth Engine to obtain Landsat imagery data for a specific area and then applies our B-CGAP method for pixel classification. The tool is very easy and efficient to use, allowing for flexible control over the selected area and time.

In terms of future directions, we suggest two main areas for improvement. The first area focuses on augmenting the datasets used for training. Currently, the scarcity of annotations for urban areas, such as cities and buildings, leads to misclassification of these regions as sediment rather than land. Expanding datasets like RiverPIXELS to include more comprehensive information about urban areas or employing automatic annotation methods to generate pseudo-labels could help strengthen model performance in these challenging scenarios. Furthermore, it is imperative to expand the applicability of the models to encompass a broader range of data modalities beyond the current reliance on Landsat data. Researchers

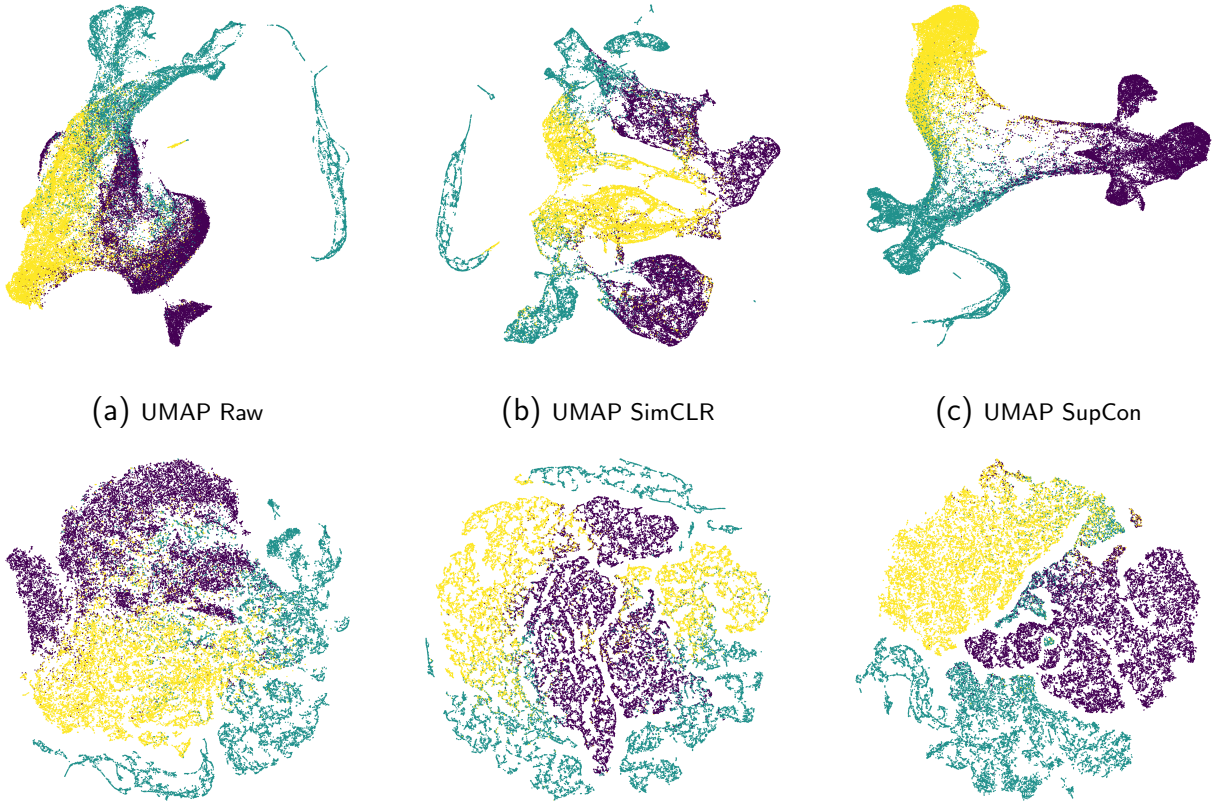
need to develop techniques to align data from a diverse range of sensors, scales, and channel characteristics to ensure the effective application of the proposed methods.

The second area for future work concerns the development of improved algorithms for the feature learning and active learning components of the proposed pipeline. Firstly, one could consider incorporating transfer learning by leveraging newly acquired unlabeled images to refine our feature-embedding neural network using the SimCLR loss. Secondly, one could explore novel graph-based semi-supervised learning methods to improve the classifier performance within our pipeline. Methods such as p-Laplace learning [Cal18, FCL22] or reweighting the graph before processing the Laplace learning [CS20, MC23] could provide further improvement.



(a) RGB (b) GT (c) B-CGAP (d) GAP (e) SVM (f) RF

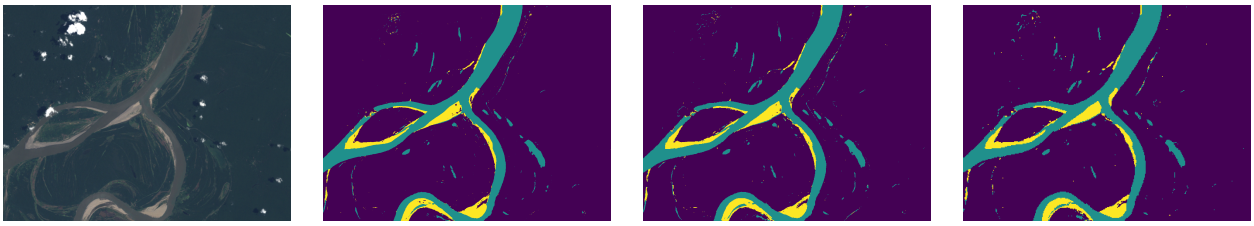
Figure 4.8: Results from a Patch of the Ucayali River. Original Patch name: "Ucayali_River_1 2018-09-11 006 066 L8 316 landsat". This patch includes some light clouds. Purple, cyan, and yellow represent land, water, and sediment respectively.



(a) UMAP Raw (b) UMAP SimCLR (c) UMAP SupCon

(d) t-SNE Raw (e) t-SNE SimCLR (f) t-SNE SupCon

Figure 4.9: This figure shows low-dimensional visualizations of feature vectors with UMAP (panels (a)-(c)) and t-SNE (panels (d)-(f)). Three columns are about raw, SimCLR, and SupCon feature vectors of pixels in the whole RiverPIXELS dataset. Purple, cyan, and yellow represent land, water, and sediment respectively.



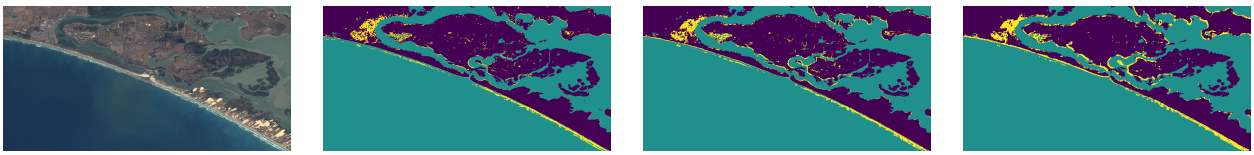
(a) RGB image. Resolution 30m. Size 558×743.

(b) Our prediction. Resolution 30m. Size 1114×1485.

(c) Our prediction. Resolution 30m. Size 558×743.

(d) Our prediction. Resolution 30m. Size 279×372.

Figure 4.10: Results from an image of the **Ucayali River** which is not included in the RiverPIXELS dataset. Region information: a rectangle centering at -73.4487, -4.45291 with a longitude range of 0.2 and a latitude range of 0.15. Panel (a) is the RGB visualization of the 30-meter resolution. Panels (b) - (d) are three predictions of the same region with different resolutions. Purple, cyan, and yellow represent land, water, and sediment respectively. Our predictions are robust among various resolutions.



(a) RGB image. Resolution 30m. Size 557×1114.

(b) Our prediction. Resolution 30m. Size 1115×2228.

(c) Our prediction. Resolution 30m. Size 557×1114.

(d) Our prediction. Resolution 30m. Size 278×557.

Figure 4.11: Results from an image of the **Murray River** which is not included in the RiverPIXELS dataset. Region information: a rectangle centering at 138.88, -35.559 with a longitude range of 0.3 and a latitude range of 0.15. Panel (a) is the RGB visualization of the 30-meter resolution. Panels (b) - (d) are three predictions of the same region with different resolutions. Purple, cyan, and yellow represent land, water, and sediment respectively. Our predictions are robust among various resolutions.

CHAPTER 5

Graph-based Active Learning for Nearly Blind Hyperspectral Unmixing

Hyperspectral unmixing serves as a powerful tool for determining the material composition of each pixel in a hyperspectral image, which typically contains hundreds of spectral channels. In this chapter, we introduce two novel graph-based semi-supervised unmixing methods. The first method directly applies graph learning to the unmixing problem, while the second method solves an optimization problem that integrates the linear unmixing model with a graph-based regularization term. Adhering to a semi-supervised framework, our methods necessitate only a minimal number of training pixels, which can be selected using a graph-based active learning approach. We assume that the ground truth information at these selected pixels can be obtained, either in the form of the exact abundance value or the one-hot pseudo label. In practical applications, the latter is considerably easier to acquire and can be achieved by minimally involving a human in the loop. In comparison to other widely used blind unmixing methods, our methods substantially enhance performance with minimal supervision. Specifically, the experimental results demonstrate that the proposed methods outperform state-of-the-art blind unmixing approaches by 50% or more, utilizing only 0.4% of training pixels.

5.1 Introduction

Data obtained by hyperspectral sensors provide both spatial and spectral representations of a scene. Compared to regular color images, which only have three color channels (Red, Green, and Blue), hyperspectral images often contain hundreds to thousands of spectral channels. However, hyperspectral imaging is limited by its low spatial resolution, and hence hyperspectral unmixing (HSU) is an effective tool to identify the pure materials and estimate the proportions of constituent endmembers at each pixel, also known as the *abundance map*. The spectral signature of a pure material is called *endmember*, which can often be measured under a laboratory setting. Unfortunately, the ground-truth endmember is often unavailable due to its large variability in any real scenario. The blind unmixing process involves the estimation of all the endmembers and the abundance map simultaneously.

5.1.1 Literature Review of HSU

In our study, we employ a *linear* mixing model for unmixing, wherein each pixel's spectral measurement is represented as a linear combination of constituent endmembers. Given the physical interpretation of hyperspectral mixing, we impose nonnegativity constraints on both endmembers and the abundance map. Additionally, we apply a sum-to-one constraint, a common practice in HSU, signifying that each pixel's abundance vector resides within the probability simplex. There are some extended linear mixing models that consider endmember-wise scaling factors [DVH16] and the spectral variability [HYC19b]. Note that these nonlinear mixing models [HPG14, BPD12] rely on more complicated assumptions about how light rays interact with endmembers. It is also plausible to remove the sum-to-one constraint when illumination conditions or the topography of the scene change locally [DVH16].

Specifically for blind HSU, it is natural to apply the nonnegative matrix factorization (NMF) [LS99, FLW22] that decomposes the data matrix into a product of two matrices with

nonnegative entries (one encodes the endmember matrix and the other is the abundance map) [SDB03, PPP06, CZP09]. However, even with the nonnegativity and sum-to-one constraints, blind HSU is a highly ill-posed inverse problem, and hence a variety of regularizations have been proposed to refine the solution space. One classic method is the ℓ_2-norm in fully constrained least squares unmixing (FCLSU) [Hei01]. Furthermore, spatial sparsity of abundances is a reasonable assumption due to the fact that only a few endmembers could appear in a single pixel. Some popular sparsity-promoting regularizations used in HSU include the ℓ_0-norm [IBP11], the ℓ_1-norm [HZZ16], the $\ell_{1/2}$-norm [QJZ11], and the mixed $\ell_{p,q}$-norm for group sparsity [DMC19]. By treating the abundance map for each material as an image, total variation (TV) regularization [ROF92] has been applied to HSU for spatial continuity and edge preservation. TV-related approaches include sparse unmixing via variable splitting augmented Lagrangian and total variation (SUnSAL-TV) [IBP12], TV with sparse NMF [HZZ17], TV with nonnegative tensor factorization [XQZ18], and an improved collaborative NMF with TV (ICoNMF-TV) [YZW20]. Recently, TV is reformulated as a quadratic regularization that promotes the minimum volume in the NMF framework, referred to as QMV [ZLF19].

Graph-based approaches [BM19] also play an important role in HSU. TV has been extended from vectors in Euclidean space to signals defined on a graph. For example, the graph TV (gTV) [BKB16] is a special case of the p-Dirichlet form [SNF13, SM14] in graph signal processing. Some graph regularization techniques for hyperspectral imaging include structured sparse regularized NMF (SS-NMF) [ZWX14] and graph-regularized $\ell_{1/2}$-NMF (GLNMF) [LWY12]. Graph-based approaches, while powerful, can suffer from intensive computation, particularly when computing pairwise similarity between pixels. Strategies in speeding up the weight computation include the use of superpixels [LZG19] rather than using the entire hyperspectral image and the Nyström method [FBC04] to generate low-rank approximations of the graph Laplacian [QLC19, QLC21]. Another efficient alternative is the use of sparse weight matrices, such as the K-Nearest Neighbors (KNN) weight matrix

[AMN98].

In recent years, neural networks have been applied to the blind unmixing problem, such as two-staged self-supervised networks [OKA18, DB21], a minimal simplex convolutional neural network [RKS22], a two-stream Siamese deep network [HGY21], and attention networks [HHG22]. Furthermore, some semi-supervised advanced deep learning methods [HYG19, HYC19a] have integrated the use of the graph Laplacian and exhibited remarkable potential in HSU. None of these methods address the issues discussed in the next paragraph.

5.1.2 Motivation and Our Contributions

There are limitations of existing blind HSU methods in many real-world scenarios. For example, unmixing often requires estimation of the number of endmembers [VBC20], while the abundance maps require human experts to convey meaning to each endmember. Meanwhile, acquiring the ground-truth abundance maps or endmember spectra is a great challenge [BJ20, ZWZ22], making it extremely difficult to train fully supervised models. Such a drawback motivates us to consider a semi-supervised model with pseudo labels. These are representative pixels for each endmember, which can be easily obtained by expert's visual inspection of the data. Prior knowledge of the number of endmembers is naturally included in the pseudo labels. We further adopt an active learning [MLB20, MMS22, MB24] approach to reinforce semi-supervised machine learning methods by carefully automating the selection its training set. Active learning has been successfully applied to hyperspectral image classification and segmentation tasks [RGC08, LBP10, MM19, CYX20]. The core of active learning is to sample data points according to an acquisition function [JH12, QSW19, MMS22] that automates the introduction of new training data during the algorithm. We believe active learning and pseudo labels would be an ideal combination to improve the performance of HSU.

The key problem we solve in this chapter is to maximize the improvement of our estimated abundance maps and endmember spectra using minimal supervision. With the training pixels

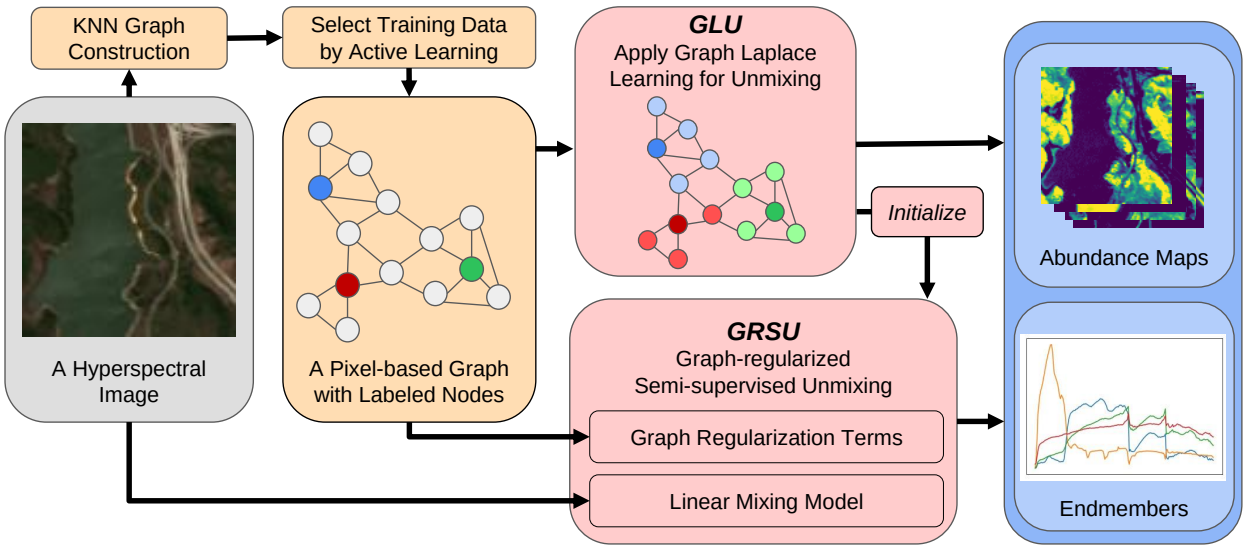


Figure 5.1: The flowchart of our semi-supervised hyperspectral unmixing models. The gray box indicates an input hyperspectral image. The orange boxes are the graph construction and graph-based active learning to select labeled nodes (pixels) for the training process (Section 2.2.1, 2.3, and 5.2.1). Two red boxes are our proposed models, Graph Learning Unmixing, GLU, (Section 5.2.2) and Graph-regularized Semi-Supervised Unmixing, GRSU, (Section 5.2.3). GLU applies graph Laplace learning directly to the unmixing task while GRSU combines the graph-based regularization term with the linear unmixing model from hyperspectral imaging into a joint optimization to be solved. GLU also serves to initialize the GRSU optimization process. The blue boxes are the outputs of GLU and GRSU, i.e., estimated endmembers and abundance map.

selected by active learning, we propose two semi-supervised hyperspectral unmixing models. We refer to these methods as *nearly blind hyperspectral unmixing*, since both of them require a very small number of training labels, and accept either pseudo one-hot labels or ground-truth abundance maps. Our first model, called graph learning unmixing (GLU), takes the output of a graph learning method [BM19] directly as the abundance maps for the hyperspectral unmixing, followed by the estimation of the endmember matrix. Our second model, called graph-regularized semi-supervised unmixing (GRSU), combines the linear unmixing model,

a graph-based regularization term, and a loss function applied to the label set (obtained by active learning) by solving a joint optimization problem. The flowchart of our proposed models (GLU and GRSU) is illustrated in Figure 5.1. We summarize the novelties of as follows,

1. We present an effective pipeline (Section 5.2.1) to select labeled pixels by graph-based active learning for the hyperspectral unmixing problem.

2. We apply the idea of graph Laplace learning to the hyperspectral unmixing problem by taking its output (class probability) as the estimated abundance map. Based on this idea, we develop the GLU model (Section 5.2.2).

3. We develop a novel semi-supervised hyperspectral unmixing model, GRSU (Section 5.2.3), by combining graph-based regularization terms with the linear mixing model and a small number of labeled pixels.

4. Our proposed methods, GLU and GRSU, bear significant practical implications. By utilizing only a small number of easily obtainable pseudo labels, our methods markedly improve the HSU performance.

All codes of our proposed methods and following experiments are available on our Github repository[1].

5.2 Semi-supervised Hyperspectral Unmixing

This section details our semi-supervised hyperspectral unmixing methods based on the linear mixing model. Specifically given a hyperspectral data cube I of the dimension $m \times n \times p$ with p spectral channels, we reshape I into a matrix $X^{\text{mat}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{p \times N}$, where $N = m \times n$ is the number of pixels. Denote a nonnegative constraint set $\Omega_{p \times q} = \{S \in \mathbb{R}^{p \times q} :$

[1] `https://github.com/wispcarey/Nearly-Blind-Hyperspectral-Unmixing`

$S_{ij} \geq 0\}$ and a probability simplex constraint set $\Pi_{q\times N} = \{A \in \Omega_{q\times N} : \mathbf{1}_q^\top A = \mathbf{1}_N^\top\}$. We assume a linear mixing model that generates the data, i.e.,

$$X^{\text{mat}} = SA + E, \tag{5.1}$$

where $S \in \Omega_{p\times q}$ is the endmember spectrum matrix, $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N] \in \Pi_{q\times N}$ is the matrix of abundance maps of q materials, and the matrix $E \in \mathbb{R}^{p\times N}$ denotes a noise term.

In Section 5.2.1, we describe the training data selection process by adapting the graph-based active (Section 2.3) to the hyperspectral setting. Then, in Section 5.2.2, we introduce the graph learning unmixing (GLU) model, which applies graph Laplace learning (Section 2.2.2) directly to the HSU problem. Lastly, in Section 5.2.3, we propose our graph-regularized semi-supervised unmixing (GRSU) model that combines graph-based regularization terms with the linear mixing model (5.1).

5.2.1 Training Data Selection

For the reshaped hyperspectral matrix $X^{\text{mat}} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{p\times N}$, let X denote the corresponding feature vector set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^p$. Given X, we can construct a graph $G = (X, W)$ according to Section 2.2.1. Then we apply the graph-based active learning process (Section 2.3) to select a set of pixels to acquire labels for both of the proposed models, GLU (Section 5.2.2) and GRSU (Section 5.2.3).

Given two positive integers $m < M$, we begin the active learning process with an initial label set of m random pixels. In each iteration, we apply graph Laplace learning (Section 2.2.2) with the current label set and calculate the acquisition function on the remaining pixels. Based on the values obtained by the acquisition function, we select a query set to be augmented to the label set and terminate this iterative process when the size of the current label set reaches M. Algorithm 4 summarizes the active learning process and its outputs, the label dataset $X_l = \{\mathbf{x}_{l_1}, \mathbf{x}_{l_2}, \ldots, \mathbf{x}_{l_M}\} \subset X$ and the set of corresponding labels $Y_l^\dagger = \{\mathbf{y}_{i_1}^\dagger, \mathbf{y}_{i_2}^\dagger, \ldots, \mathbf{y}_{i_M}^\dagger\}$, serve as the training data for our semi-supervised unmixing

Algorithm 4 Sample Labeled Pixels through Active Learning

Require: Dataset X; corresponding graph $G = (X, W)$; initial sample number m; total sample number M.

Ensure: The label dataset $X_l \subset \mathcal{X}$ and the set of corresponding labels $Y_l^\dagger$.

1: ***Initialize:*** Randomly sample m pixels as the initial label set X_l; acquire the labels for X_l as $Y_l^\dagger$.

2: **while** $|X_l| < M$ ($|\cdot|$ means set cardinality) **do**

3: Apply the graph Laplace learning on G based on labels of X_l to predict labels on $X_u = X \setminus X_l$.

4: Calculate the acquisition function $\mathcal{A}$ on X_u based on the Laplace learning outputs.

5: Select a query set $\mathcal{Q}$ based on the acquisition function values according to the sequential active learning or LocalMax.

6: Update the current label set: $X_l \to X_l \cup \mathcal{Q}$; Acquire the labels of $\mathcal{Q}$ and update the label set $Y_l^\dagger$ accordingly.

7: **end while**

framework.

We want to clarify three aspects of the outputs of active learning (Algorithm 4). First, we should acquire labels in Algorithm 4 through an oracle or a human-in-the-loop process. Second, the ground-truth "label" $\mathbf{y}_i^\dagger \in \mathbb{R}^{q\times 1}$ for the "labeled" pixel $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$ can be either the ground-truth abundance vector or its one-hot pseudo label, the latter of which can be determined by the experts for identifying the most significant endmember. According to the experimental results in Section 5.3, requiring the ground-truth abundance for active learning is unnecessary. Third, we adopt the matrix forms of $X_l^{\text{mat}} = [\mathbf{x}_{l_1}, \mathbf{x}_{l_2}, \ldots, \mathbf{x}_{l_M}] \in \mathbb{R}^{p\times M}$, $A_L = [\mathbf{y}_1^\dagger, \mathbf{y}_2^\dagger, \ldots, \mathbf{y}_M^\dagger] \in \mathbb{R}^{q\times M}$ of the output sets X_l and $Y_l^\dagger$, respectively.

Given the training matrix X_l^{mat}, we estimate the abundance map A for the entire data matrix X^{mat}. There is indeed an overlap between X_l^{mat} and X^{mat}, but we cannot wholly trust the training labels, especially those obtained by using the one-hot pseudo labels, because

they are not the abundance values we aim to estimate. As a result, we update the abundance map for all the pixels even though a subset of them is selected to acquire some sort of ground-truth information. Another rationale for having two separate matrices X^{mat} and X_l^{mat} is the option to select the training pixels from one image and perform semi-supervised unmixing on the other image, which falls out of the scope of the application discussed in this chapter.

5.2.2 Graph Learning Unmixing (GLU)

Following the training data selection process, we obtain a training data matrix $X_l^{\text{mat}} \in \mathbb{R}^{p\times M}$ consisting of M labeled pixels. We concatenate X_l^{mat} with the original data matrix X^{mat}, and construct a graph $\widetilde{G}$ based on the combined data matrix $\widetilde{X} = [X_l^{\text{mat}}, X^{\text{mat}}]$ with the corresponding graph Laplacian $\widetilde{L}$ according to Section 2.2.1. The graph Laplace learning produces the class probability, which can be regarded as the abundance map. Specifically, we estimate the abundance map by projecting the graph Laplace learning solution A_{GL} onto $\Pi_{q\times N}$, i.e.,

$$A_{\text{GL}} = \underset{A\in\mathbb{R}^{q\times N}}{\arg\min} \frac{1}{2}\left\langle [A_L, A]^\top, \widetilde{L}[A_L, A]^\top \right\rangle_F, \tag{5.2}$$

$$A_{\text{GLU}} = \mathcal{P}_{\Pi_{q\times N}}(A_{\text{GL}}). \tag{5.3}$$

Problem (5.2) is the standard graph Laplace learning. Consider the block representation of $\widetilde{L}$ as

$$\widetilde{L} = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix} \in \mathbb{R}^{(M+N)\times(M+N)}, \tag{5.4}$$

where $L_{ll} \in \mathbb{R}^{M\times M}$ and $L_{uu} \in \mathbb{R}^{N\times N}$ are the parts corresponding to the labeled pixel set X_l and the unlabeled pixel set (indeed the whole set) X, respectively, and $L_{ul} = L_{lu}^\top \in \mathbb{R}^{N\times M}$ represents the cross interaction of graph Laplacian between X_l and X. Then the minimization problem (5.2) has a closed-form solution

$$A_{\text{GL}} = -A_L L_{lu} L_{uu}^{-1}, \tag{5.5}$$

which can be solved by the preconditioned conjugate gradient method thanks to the symmetric and semi-positive definite properties of the part L_{uu}. Equation (5.3) is to project the output $A_{\mathrm{GL}} \in \mathbb{R}^{q\times N}$ of the graph Laplace learning onto the set $\Pi_{q\times N}$ by a projection operator $\mathcal{P}_{\Pi_{q\times N}}$, defined by,

$$\mathcal{P}_{\Pi_{q\times N}}(A) = \underset{V\in\Pi_{q\times N}}{\arg\min} \|V - A\|_F. \tag{5.6}$$

This projection operator can be implemented by a fast algorithm [WC13].

Given A_{GLU}, we can then find the optimal endmember matrix S_{GLU} that minimizes the combination of the least-squares errors of the linear mixing model (5.1) and the misfit of the training data, i.e.,

$$S_{\mathrm{GLU}} = \underset{S\in\Omega_{p\times q}}{\arg\min} \frac{1}{2}\|X^{\mathrm{mat}} - SA_{\mathrm{GLU}}\|_F^2 + \frac{\alpha^2}{2}\|X_l^{\mathrm{mat}} - SA_L\|_F^2, \tag{5.7}$$

with a weighting parameter $\alpha > 0$. Equation (5.7) has a closed-form solution

$$S^0_{\mathrm{GLU}} = \left(X^{\mathrm{mat}}A^\top + \alpha^2 X_l^{\mathrm{mat}}A_L^\top\right)\left(AA^\top + \alpha^2 A_L A_L^\top\right)^{-1}, \tag{5.8}$$

$$S_{\mathrm{GLU}} = \max\left(0, S^0_{\mathrm{GLU}}\right). \tag{5.9}$$

Equation (5.9) means to take the entry-wise maximum of S^0_{GLU} and 0, i.e., replace each negative entry in S^0_{GLU} by 0. Algorithm 5 presents the pseudo-code of our GLU method, which only involves three steps to find A_{GLU} and S_{GLU} (no iteration is needed).

5.2.3 Graph-regularized Semi-supervised Unmixing (GRSU)

By assuming the linear mixing model (5.1), it is standard to solve the blind unmixing problem in a regularized least square form,

$$\underset{\substack{S\in\Omega_{p\times q},\\ A\in\Pi_{q\times N}}}{\arg\min} \frac{1}{2}\|X^{\mathrm{mat}} - SA\|_F^2 + \lambda\mathcal{J}(A), \tag{5.10}$$

with a positive weighting parameter λ. The term $\frac{1}{2}\|X^{\mathrm{mat}} - SA\|_F^2$ is a least-squares misfit between the matrix product SA and the data measurement X^{mat}, while $\mathcal{J}(A)$ is a regular-

Algorithm 5 Graph Learning Unmixing (GLU)

Require: Data matrix X^{mat}, training data (X_L^{mat}, A_L), and $\alpha > 0$.

Ensure: Matrices S_{GLU} and A_{GLU}.

1: ***Initialize:*** Build a graph on $\widetilde{X} = [X_L^{\text{mat}}, X^{\text{mat}}]$ with corresponding Laplacian matrix $\widetilde{L}$. Segment $\widetilde{L}$ into the block form (5.4).

2: **Graph Learning Step:**

3: $A_{\text{GL}} = -A_L L_{lu} L_{uu}^{-1}$.

4: **Projection:**

5: $A_{\text{GLU}} = \mathcal{P}_{\Pi_{q\times N}}(A_{\text{GL}})$.

6: **Estimate the endmember spectrum matrix:**

7: $S_{\text{GLU}}^0 = (X^{\text{mat}} A^\top + \alpha^2 X_L^{\text{mat}} A_L^\top)(AA^\top + \alpha^2 A_L A_L^\top)^{-1}$,

8: $S_{\text{GLU}} = \max(0, S_{\text{GLU}}^0)$.

ization term of the abundance matrix A. We impose the nonnegative constraints of S and A as well as a sum-to-one constraint of A.

Following the graph Dirichlet energy [OWO14, Eva22], we consider a graph Laplacian regularization, formulated by

$$\mathcal{J}_1(A) = \frac{1}{4}\sum_{i,j=1}^{N} \|\mathbf{a}_i - \mathbf{a}_j\|_2^2 w(\mathbf{x}_i, \mathbf{x}_j), \tag{5.11}$$

where the weight function $w : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is defined by

$$w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\angle(\mathbf{x}_i, \mathbf{x}_j)^2}{\sqrt{\tau_i \tau_j}}\right),$$

with the angular distance $\angle(\mathbf{x}_i, \mathbf{x}_j)$ between node $\mathbf{x}_i$ and $\mathbf{x}_j$ and τ_i, τ_j are defined in Section 2.2.1.

In addition, we further develop a semi-supervised graph regularization term $\mathcal{J}_2(A; A_L, X_l^{\text{mat}})$ that includes the label information X_l^{mat}, A_L, that is,

$$\mathcal{J}_2(A; A_L, X_l^{\text{mat}}) = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{M} \|\mathbf{a}_i - \mathbf{y}_j^\dagger\|_2^2 w(\mathbf{x}_i, \mathbf{x}_{l_j}). \tag{5.12}$$

We consider the sum of both terms $\mathcal{J}_1(A)$ and $\mathcal{J}_2(A; A_L, X_l^{\text{mat}})$ as the regularization $\mathcal{J}(A)$. As $\mathcal{J}_1$ and $\mathcal{J}_2$ have the same form that leads to the same scale, we assign the equal weight of them when formulating $\mathcal{J}$, i.e., $\mathcal{J}(A) = \mathcal{J}_1(A) + \mathcal{J}_2(A; A_L, X_l^{\text{mat}})$.

Putting (5.10)–(5.12) together with the label information (X_l^{mat}, A_L) obtained by the active learning approach, we propose a graph-regularized semi-supervised unmixing (GRSU) model to simultaneously estimate the abundance map A_{GRSU} and the endmember matrix S_{GRSU}, that is,

$$S_{\text{GRSU}}, A_{\text{GRSU}} = \underset{\substack{S\in\Omega_{p\times q},\\ A\in\Pi_{q\times N}}}{\arg\min} \frac{1}{2}\|X^{\text{mat}} - SA\|_F^2 + \frac{\alpha^2}{2}\|X_l^{\text{mat}} - SA_L\|_F^2 + \lambda\mathcal{J}_1(A) + \lambda\mathcal{J}_2(A; A_L, X_l^{\text{mat}}), \tag{5.13}$$

with two positive parameters α and λ. We define the indicator function

$$\chi_\Delta(Z) = \begin{cases} 0, & Z \in \Delta, \\ \infty, & \text{otherwise}, \end{cases} \tag{5.14}$$

to rewrite the minimization problem (5.13) into an unconstrained formulation as follows,

$$\min_{A,S} \ \frac{1}{2}\|X^{\text{mat}} - SA\|_F^2 + \frac{\alpha^2}{2}\|X_l^{\text{mat}} - SA_L\|_F^2 + \lambda\mathcal{J}_1(A) + \lambda\mathcal{J}_2(A; A_L, X_l^{\text{mat}}) + \chi_{\Omega_{p\times q}}(S) + \chi_{\Pi_{q\times N}}(A). \tag{5.15}$$

We apply the alternating direction method of multipliers (ADMM) [EB92] to solve the unconstrained problem (5.15). In particular, we introduce two auxiliary variables $T \in \mathbb{R}^{p\times q}$ and $B \in \mathbb{R}^{q\times N}$ to express the problem (5.15) equivalently as

$$\begin{aligned} \min_{A,B,S,T} \ & \frac{1}{2}\|X^{\text{mat}} - TA\|_F^2 + \frac{\alpha^2}{2}\|X_l^{\text{mat}} - TA_L\|_F^2 + \lambda\mathcal{J}_1(B) \\ & + \lambda\mathcal{J}_2(B; A_L, X_l^{\text{mat}}) + \chi_{\Omega_{p\times q}}(S) + \chi_{\Pi_{q\times N}}(A) \\ \text{s.t.} \quad & A = B, S = T. \end{aligned} \tag{5.16}$$

The augmented Lagrangian of (5.16) is written as

$$\begin{aligned}\mathcal{L} = \frac{1}{2}\|X^{\text{mat}} - TA\|_F^2 + \frac{\alpha^2}{2}\|X_l^{\text{mat}} - TA_L\|_F^2 + \lambda\mathcal{J}_1(B) \\ + \lambda\mathcal{J}_2(B; A_L, X_l^{\text{mat}}) + \chi_{\Omega_{p\times q}}(S) + \chi_{\Pi_{q\times N}}(A) \\ + \frac{\rho}{2}\|A - B + \bar{B}\|_F^2 + \frac{\gamma}{2}\|S - T + \bar{T}\|_F^2,\end{aligned}$$

with dual variables $\bar{B}, \bar{T}$ and two positive constants ρ, γ. One benefit of ADMM is that it turns the joint minimization problem (5.16) into four subproblems that are associated $A,\ B,\ S,\ T$ separately. In each iteration, we iterate as follows,

$$\begin{aligned}
T &\leftarrow \underset{T\in\mathbb{R}^{p\times q}}{\arg\min}\ \frac{1}{2}\|[X^{\text{mat}}, \alpha X_l^{\text{mat}}] - T[A, \alpha A_L]\|_F^2 + \frac{\gamma}{2}\|S - T + \bar{T}\|_F^2, \\
S &\leftarrow \underset{S\in\Omega_{p\times q}}{\arg\min}\ \frac{\gamma}{2}\|S - T + \bar{T}\|_F^2, \\
A &\leftarrow \underset{A\in\Pi_{q\times N}}{\arg\min}\ \frac{1}{2}\|X^{\text{mat}} - TA\|_F^2 + \frac{\rho}{2}\|A - B + \bar{B}\|_F^2, \\
B &\leftarrow \underset{B\in\mathbb{R}^{q\times N}}{\arg\min}\ \lambda\mathcal{J}_1(B) + \lambda\mathcal{J}_2(B, A_L, X_l^{\text{mat}}) + \frac{\rho}{2}\|A - B + \bar{B}\|_F^2, \\
\bar{B} &\leftarrow \bar{B} - A + B, \\
\bar{T} &\leftarrow \bar{T} - S + T.
\end{aligned} \tag{5.17}$$

The A, S, and T - subproblems are the same as in the blind unmixing paper [QLC19], thus the details are omitted.

As for the B-subproblem, we write it explicitly by using $B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]$,

$$\begin{aligned}
B =& \underset{B\in\mathbb{R}^{q\times N}}{\arg\min}\ \mathcal{J}_1(B) + \mathcal{J}_2(B, A_L, X_l^{\text{mat}}) + \frac{\rho}{2\lambda}\|A - B + \bar{B}\|_F^2 \\
=& \underset{B\in\mathbb{R}^{q\times N}}{\arg\min}\ \frac{1}{4}\sum_{i,j=1}^{N}\|\mathbf{b}_i - \mathbf{b}_j\|_2^2 w(\mathbf{x}_i, \mathbf{x}_j) \\
&+ \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{M}\|\mathbf{b}_i - \mathbf{y}_j^\dagger\|_2^2 w(\mathbf{x}_i, \mathbf{x}_{l_j}) \\
&+ \frac{1}{4}\sum_{i,j=1}^{M}\|\mathbf{y}_i^\dagger - \mathbf{y}_j^\dagger\|_2^2 w(\mathbf{x}_{l_i}, \mathbf{x}_{l_j}) + \frac{\rho}{2\lambda}\|A - B + \bar{B}\|_F^2.
\end{aligned} \tag{5.18}$$

Algorithm 6 Graph-regularized Semi-supervised Unmixing (GRSU)

Require: Data matrix X^{mat}, label information (X_L^{mat}, A_L), parameters $\alpha, \lambda, \gamma, \rho$, maximum iteration $I_{\max}$, and error tolerance ϵ.

Construct: A graph $\widetilde{G}$ on $\widetilde{X} = [X_L^{\text{mat}}, X^{\text{mat}}]$ (by Section 2.2.1) with the graph Laplacian $\widetilde{L}$ and its block form of (5.4)

Initialize: $S^0 = S_{\text{GLU}}$ and $A^0 = A_{\text{GLU}}$ (by GLU Algorithm 5); $B^0 = A^0$, $\bar{B}^0 = 0$, $\bar{T} = 0$, Err $= 1$, and $i = 0$.

while $i < I_{\max}$ **and** Err $> \epsilon$ **do**

$T^{i+1} = (X^{\text{mat}}(A^i)^\top + \alpha^2 X_L^{\text{mat}}(A_L)^\top + \gamma(S^i + \bar{T}^i))(A^i(A^i)^\top + \alpha^2 A_L(A_L)^\top + \gamma I_q)^{-1}$.

$S^{i+1} = \max(T^{i+1} - \bar{T}^i, 0)$.

$A^{i+1} = \mathcal{P}_{\Pi_{q\times N}}\left(((S^{i+1})^\top S^{i+1} + \rho I_q)^{-1}((S^{i+1})^\top X^{\text{mat}} + \rho(B^i - \bar{B}^i)\right)$.

$B^{i+1} = (-A_L L_{ul} + \frac{\rho}{\lambda}(A^{i+1} + \bar{B}^i))(L_{uu} + \frac{\rho}{\lambda} I_N)^{-1}$

$\bar{B}^{i+1} = \bar{B}^i + (A^{i+1} - B^{i+1})$.

$\bar{T}^{i+1} = \bar{T}^i + (S^{i+1} - T^{i+1})$.

$i \leftarrow i + 1$ and Err $= \max((\|S^{i+1} - S^i\|_F)/(\|S^i\|_F), (\|A^{i+1} - A^i\|_F)/(\|A^i\|_F))$

end while

Ensure: $S_{\text{GRSU}} = S^i$, $A_{\text{GRSU}} = A^i$

Note that we add the term $\frac{1}{4}\sum_{i,j=1}^{M} \|\mathbf{y}_i^\dagger - \mathbf{y}_j^\dagger\|_2^2 w(\mathbf{x}_{l_i}, \mathbf{x}_{l_j})$ in (5.18) that does not affect the minimization over B, but rather turns the B-subproblem (5.18) into a regularized graph Laplacian learning problem (see Section 2.2.2). Specifically, we consider the graph $\widetilde{G}$ built on the combined data matrix $\widetilde{X} = [X_l^{\text{mat}}, X^{\text{mat}}]$ with the graph Laplacian matrix $\widetilde{L}$. Then (5.18) is equivalent to

$$B = \underset{B\in\mathbb{R}^{q\times N}}{\arg\min} \frac{1}{2}\left\langle [A_L, B]^\top, \widetilde{L}[A_L, B]^\top \right\rangle_F + \frac{\rho}{2\lambda}\|A - B + \widetilde{B}\|_F^2. \tag{5.19}$$

Using the block representation of $\widetilde{L}$ in (5.4), we have a closed-form solution to (5.19) as

$$B = \left(-A_L L_{lu} + \frac{\rho}{\lambda}(A + \widetilde{B})\right)\left(L_{uu} + \frac{\rho}{\lambda} I_N\right)^{-1}, \tag{5.20}$$

where I_N is an $N \times N$ identity matrix.

The semi-supervised unmixing (GRSU) method is summarized in Algorithm 6. Its initial values of A^0 and S^0 are obtained by GLU (Section 5.2.2) that outputs $A_{\text{GLU}}, S_{\text{GLU}}$.

5.3 Experiments and Results

In this section, we conduct extensive experiments to demonstrate the performance of the proposed unmixing models. Specifically, in Section 5.3.1, we compare our semi-supervised methods (GLU and GRSU) with the state-of-the-art (unsupervised) blind unmixing methods, followed by a discussion of our semi-supervised unmixing methods with respect to different numbers of training pixels in Section 5.3.2. In Section 5.3.3, we test the robustness of various methods by adding different amounts of Gaussian white noise to the HSI. We test on four standard hyperspectral image datasets, described as follows,

1. Jasper Ridge: The Jasper Ridge dataset [Has] is a hyperspectral image of the size 100×100 with 198 channels. Originally it had 224 hyperspectral channels spanning from 380 to 2500 nm, and 26 channels are removed as a preprocessing step due to dense water vapor and atmospheric effects. Four endmembers are latent: Tree, Water, Dirt, and Road.

2. Samson: The Samson dataset [Has] is of the size 95×95 with 156 channels that span from 401 to 889 nm. Three endmembers are latent: Soil, Tree, and Water. Note that a different ground truth is considered in [RKS22], while we use the original ground-truth information.

3. Urban4: The Urban dataset [Has] is of the size 307×307 with 162 channels. Each pixel of this image corresponds to a $2 \times 2m^2$ area. The original 221 channels span from 400 nm to 2500 nm. There are three versions of the ground truth, which contain 4, 5, and 6 endmembers. Here we use the version of four endmembers, labeled as Asphalt, Grass, Tree, and Roof.

4. Apex: The Apex dataset[2] [SJH15] is a hyperspectral image of the size 111×122 with 285 bands spanning from 413 to 2420nm. Four endmembers are latent in this data: Road, Tree, Roof, and Water.

We apply two metrics, root mean square error (RMSE) and spectral angle distance (SAD), to evaluate the quality of the abundance matrix A and the endmember spectrum matrix S, respectively. RMSE and SAD are defined as follows,

$$\mathrm{RMSE}(A, A^{\mathrm{gt}}) = 100 \times \sqrt{\frac{1}{pN}\|A - A^{\mathrm{gt}}\|_F^2} \tag{5.21}$$

$$\mathrm{SAD}(S, S^{\mathrm{gt}}) = \frac{180}{\pi} \times \frac{1}{p}\sum_{i=1}^{p} \arccos\left(\frac{\langle \mathbf{s}_i, \mathbf{s}_i^{\mathrm{gt}}\rangle}{\|\mathbf{s}_i\|_2 \|\mathbf{s}_i^{\mathrm{gt}}\|_2}\right), \tag{5.22}$$

where A, S are the fitted matrices, $A^{\mathrm{gt}}, S^{\mathrm{gt}}$ are the ground-truth, and $\mathbf{s}_i$ denotes the i^{th} column of the matrix S.

5.3.1 Method Comparison

We compare our semi-supervised methods with five state-of-the-art unsupervised unmixing methods, namely, QMV [ZLF19], GTVMBO [QLC21], MSC [RKS22] and EGU [HGY21]. The first three methods (GLNMF, QMV, GTVMBO) initialize with the output of FCLSU [Hei01]. MSC and EGU are neural network methods. Note that the EGU method supports the use of either the ground truth endmember spectrum matrix S or the estimated matrix S using vertex component analysis (VCA) [ND05]. We use VCA to estimate S, which serves an input for the pixel-wise EGU-net method. In the following experiments, GLNMF [LWY12], QMV [ZLF19], and GTVMBO [QLC21] are executed in MATLAB and conducted on an Intel i9-9900K CPU, while MSC [RKS22] and EGU [HGY21] are implemented in Python and conducted on an Nvidia 2080 Ti GPU.

For labels used in our semi-supervised framework, we consider the exact abundance map (EXT) and the one-hot pseudo label (OH). The latter (OH) can be obtained by thresholding

[2] `https://github.com/BehnoodRasti/MiSiCNet`

Training Information for Each Dataset

	Parameters				***Training Data Info***			
Dataset	α	λ	γ	ρ	*Acq Fun*	*Training pixels*	*Training Percentage*	*Num Each Class*
Jasper	10	1	1	1	MCVOPT	44	0.44%	13, 7, 16, 8
Samson	20	50	0.1	0.1	VOPT	36	0.40%	13, 14, 9
Urban4	50	500	0.1	0.1	VOPT	364	0.38%	138, 105, 72, 49
Apex	10	50	1	1	VOPT	54	0.40%	7, 24, 13, 10

Table 5.1: Parameter choices and training data information for our GLU and GRSU models. Parameters $\alpha, \lambda, \gamma, \rho$ are all associated with GRSU, while GLU only involves one parameter α (same as the one used in GRSU). "Acu Fun" means the acquisition function applied in the active learning process. "Training Pixels" means the number of labeled pixels used for the training process. "Training Percentage" means the percentage of labeled pixels to all pixels. "Num Each Class" means the number of labeled pixels of each endmember.

Computation Times for different methods

	Unsupervised Methods					***Our Semi-supervised Methods***			
Method	*GLNMF* [LWY12]	*QMV* [ZLF19]	*GTVMBO* [QLC21]	*MSC* [RKS22]	EGU [HGY21]	*GLU -OH*	*GLU -EXT*	*GRSU -OH*	*GRSU -EXT*
Jasper	8.81s^*	**2.51s***	2.77s^*	112.09s	51.36s	**2.38s**	2.45s	3.07s	2.92s
Samson	3.89s^*	1.40s^*	**0.51s***	95.32s	43.58s	1.52s	**1.47s**	10.42s	12.92s
Urban4	67.31s^*	23.57s^*	**14.60s***	974.05s	583.16s	**22.29s**	22.37s	201.89s	294.71s
Apex	26.47s^*	2.12s^*	**0.59s***	166.03s	87.84s	**3.81s**	3.97s	20.46s	24.04s

Table 5.2: This table presents the computation times of various methods applied to each dataset. Specifically, the GLNMF, QMV, and GTVMBO methods are executed in MATLAB, with their times indicated by an asterisk ($*$), while the other methods are implemented in Python. The computation time is measured in seconds. The best computation times from the unsupervised and semi-supervised methods are highlighted in bold in each row. Rhe proposed methods (GLU and GRSU) run much faster than the neural network methods approaches (MSC and EGU), and are comparable to traditional unsupervised methods (GLNMF, QMV, and GTVMBO).

RMSE between the estimated abundance matrix and the ground truth.

		Unsupervised Methods					***Ours (0.4% training)***			
Dataset	***Class***	*GLNMF* [LWY12]	*QMV* [ZLF19]	*GTVMBO* [QLC21]	*MSC* [RKS22]	EGU [HGY21]	*GLU -OH*	*GLU -EXT*	*GRSU -OH*	*GRSU -EXT*
Jasper	Tree	**10.04**	10.59	18.70	12.34	11.25	7.09	10.91	**4.04**	6.48
	Water	14.06	**4.18**	7.38	7.30	6.22	7.33	11.40	**4.39**	4.27
	Dirt	14.32	**10.93**	17.10	15.35	14.59	8.64	13.04	**6.48**	6.85
	Road	51.82	**14.74**	15.64	52.09	32.84	14.78	15.75	6.99	**6.34**
	Overall	18.79	**9.51**	15.16	18.24	13.78	8.37	12.03	**5.10**	5.93
Samson	Soil	22.51	20.28	**8.01**	20.05	30.73	4.37	5.73	5.18	**4.10**
	Tree	31.29	24.92	**16.10**	28.90	29.56	7.51	6.02	6.35	**4.38**
	Water	16.26	16.63	**9.19**	16.45	15.58	11.06	4.70	11.32	**4.89**
	Overall	25.21	21.48	**12.19**	23.32	27.08	7.81	5.61	7.66	**4.43**
Urban4	Asphalt	33.89	21.58	19.19	**18.19**	21.4	14.48	**7.99**	14.55	8.32
	Grass	**10.15**	29.07	10.44	31.61	45.83	7.61	**6.86**	7.47	6.93
	Tree	15.82	25.65	**13.87**	31.37	14.32	**7.78**	9.51	7.88	9.67
	Roof	**17.53**	31.18	23.49	38.00	18.86	9.93	10.65	**9.78**	10.59
	Overall	22.13	26.17	**15.71**	28.58	30.77	10.48	**8.30**	10.49	8.46
Apex	Road	23.13	25.34	57.78	**22.77**	43.61	11.35	17.40	**10.16**	13.68
	Tree	**13.51**	19.99	32.29	14.08	25.27	11.33	12.22	11.07	**9.44**
	Roof	29.15	25.65	46.64	**19.91**	33.67	16.67	**12.69**	17.31	13.71
	Water	16.81	8.70	21.63	**5.50**	11.56	14.85	**14.43**	15.08	14.85
	Overall	19.28	20.09	37.10	**15.27**	27.49	13.34	13.44	13.35	**12.15**

Table 5.3: Comparison results in terms of $\mathrm{RMSE}(A, A^{\mathrm{gt}})$ for the abundance maps: four of our semi-supervised methods (with around 0.4% of labeled pixels) are compared with five (unsupervised) blind unmixing methods on four publicly available datasets. For each row, the best results of unsupervised methods and our semi-supervised methods are bolded, respectively. The best of our methods achieves nearly 50% improvements over the unsupervised ones in most cases.

SAD between the estimated spectrum matrix and the ground truth.

		Unsupervised Methods					***Ours (0.4% training)***			
Dataset	***Class***	*GLNMF* [LWY12]	*QMV* [ZLF19]	*GTVMBO* [QLC21]	*MSC* [RKS22]	EGU [HGY21]	*GLU -OH*	*GLU -EXT*	*GRSU -OH*	*GRSU -EXT*
Jasper	Tree	5.63	2.66	13.95	**2.48**	8.49	7.49	2.50	**2.00**	4.70
	Water	**3.60**	5.02	22.43	16.60	14.64	22.21	22.66	**5.04**	22.27
	Dirt	6.53	**2.50**	8.37	3.80	6.68	3.99	**1.80**	1.89	2.54
	Road	43.94	**3.96**	6.61	18.88	5.16	2.41	2.05	**1.28**	2.20
	Overall	14.92	**3.55**	12.83	10.44	8.74	9.03	7.25	**2.55**	7.92
Samson	Soil	1.37	2.48	2.59	23.13	**1.35**	2.07	1.87	**1.44**	2.32
	Tree	**1.62**	3.04	5.53	2.03	2.29	4.12	**2.54**	3.20	2.56
	Water	10.48	32.94	21.40	45.95	**8.62**	9.50	30.96	**2.41**	31.45
	Overall	4.50	12.82	9.83	23.71	**4.09**	5.24	11.79	**2.36**	12.11
Urban4	Asphalt	7.36	150.64	**6.98**	43.06	7.62	6.35	**3.02**	6.25	3.08
	Grass	10.85	23.49	**6.66**	22.39	37.45	3.49	3.82	**3.35**	3.74
	Tree	9.11	7.52	**1.58**	5.67	4.86	7.02	**3.80**	6.35	4.39
	Roof	44.15	3.85	44.23	**2.59**	45.89	3.26	3.89	**3.22**	4.16
	Overall	17.87	46.37	**14.87**	18.43	23.96	5.04	**3.64**	4.79	3.83
Apex	Road	29.38	8.39	40.88	14.90	**5.61**	7.68	14.95	**4.25**	17.61
	Tree	**6.55**	11.86	24.21	7.88	7.41	10.15	**5.08**	8.03	6.62
	Roof	6.23	6.26	5.65	10.10	**4.86**	9.59	6.47	6.82	**4.30**
	Water	**9.70**	106.13	51.46	43.85	12.73	30.73	**11.33**	12.62	17.11
	Overall	12.96	33.16	30.55	19.19	**7.65**	14.54	9.46	**7.94**	11.41

Table 5.4: Comparison results in terms of $\mathrm{SAD}(S, S^{\mathrm{gt}})$ for the endmember spectrum matrices S: four of our semi-supervised methods (with around 0.4% of labeled pixels) are compared with five (unsupervised) blind unmixing methods on four publicly available datasets. For each row, the best results of unsupervised methods and our semi-supervised methods are bolded, respectively.

the exact abundance map or an expert identifying the most significant endmember, which is more practical than the former (EXT) in real circumstances.

For each experiment, the active learning process starts with only one random pixel per material as an initial label set and terminates until around 0.4% of the total pixels are sampled according to the active learning algorithm (Algorithm 4). We fix $\epsilon = 10^{-3}$, $I_{\max} = 1000$ and $K = 50$ in the KNN graph construction (Section 2.2.1). Note that K is chosen to be relatively small for computational benefits while ensuring the connectivity of the constructed graph G, which is required in the calculation of the VOpt and MCVOpt acquisition functions (2.51), (2.58). We select the optimal combination of the parameters $\alpha, \lambda, \gamma, \rho$ in the range of $\alpha \in \{10, 20, 50, 100\}$, $\lambda \in \{10^i, 5 \times 10^i | i = 0, 1, 2, 3\}$, $\gamma = \rho \in \{10^i | i = -2, -1, 0, 1, 2\}$ that yields the smallest sum of RMSE and SAD using 10% of randomly selected pixels as a validation set. We list the parameter choices, the amount of training data, and the acquisition function (Acq Fun) in the active learning approach for each dataset in Table 5.1.

Table 5.2 shows the computation times of various methods applied to each dataset. Our semi-supervised method GLU is much faster than the neural network methods approaches (MSC and EGU) and is comparable to traditional unsupervised methods (GLNMF, QMV, and GTVMBO) in terms of computation time. As the GRSU method requires solving the graph Laplace learning problem in each iteration, it does require more computation times compared to the regularization-based methods, while it is still faster than neural network methods.

Table 5.3 and Table 5.4 report the results of RMSE(A, A^{gt}) values and SAD(S, S^{gt}) values, respectively. For our methods, the "-OH" suffix refers to training on the one-hot pseudo labels, while the "-EXT" suffix refers to training on the exact abundance maps. In both tables, we highlight the best results in our four semi-supervised methods and the other five unsupervised methods, separately. By comparing the bold results in each row, the best of our methods achieves nearly 50% improvements over the competing unsupervised methods in most cases. There are only a few exceptions. For example, EGU has an outstanding

performance of the SAD on the Apex dataset, which is slightly better than ours.

In addition, we observe that the winner of the five supervised methods scatters over Table 5.3 and Table 5.4. For example, QMV and GTVMBO perform the best on Jasper Ridge and Urban4, respectively, for both abundance and endmember estimations. For the Samson dataset, GTVMBO attains the best RMSE, while EGU has the best SAD. For Apex, MSC has the best RMSE, while EGU achieves the best SAD. Our methods, on the other hand, yield consistent performance in that GRSU is generally better than GLU. One exception is the Urban4 dataset, in which GRSU-EXT is slightly worse than GLU-EXT. Since GLU is the initialization of GRSU, we can see improvements of GRSU after the ADMM iterations, which are particularly significant on the SAD values of the Jasper Ridge, Samson, and Apex datasets. Furthermore, training on one-hot pseudo labels (-OH) sometimes has a better performance than training on the exact abundance map (-EXT), which implies that it is not necessary to require the exact abundance maps for the training process of our approaches.

We arrange the estimated abundance maps and endmembers in pairs for Jasper Ridge, Samson, Urban4, and Apex datasets, sequentially, showing in Figures 5.2–5.5. Note that we normalize each endmember (column) in the spectrum matrix S to have the unit norm, i.e., $\|\mathbf{s}_i\|_2 = \|\mathbf{s}_i^{\text{gt}}\|_2 = 1$. The labeled pixels that are selected by active learning are indicated in red dots on the ground truth abundance map.

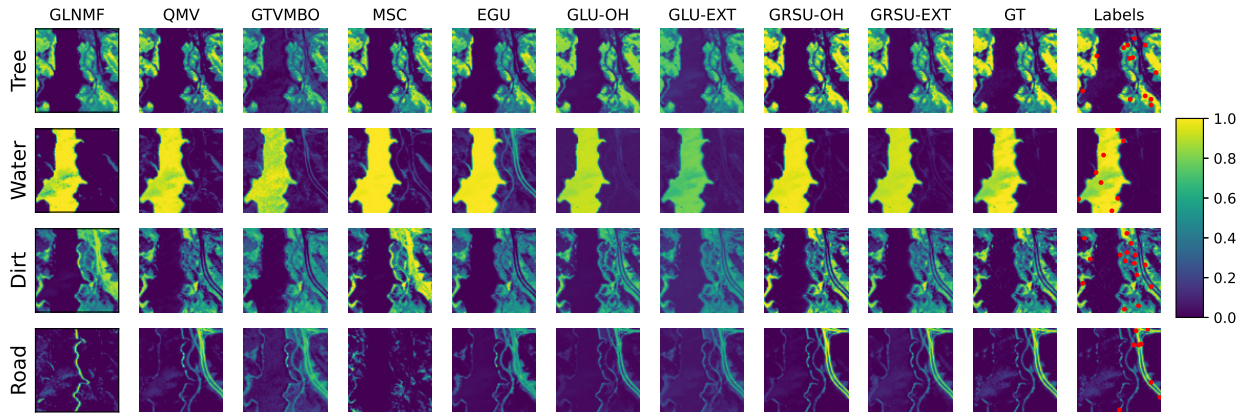
The active learning approach can identify the distribution of each endmember by selecting a few representatives. Take the Road in the Jasper Ridge dataset for an example. As illustrated in the last row of Panel(a) in Figure 5.2, the abundance map for the Road contains fine structures that are easily smeared out by other methods, while active learning can successfully identify pixels with high abundance values of this endmember to acquire labels. With those sampled labeled pixels, the estimated abundance maps' quality increases significantly compared to unsupervised methods. Another evident example is the endmember Roof in the Urban dataset (the last row of Panel (a) in Figure 5.4). Both GLU-OH and GRSU-OH well preserve the contrast of the rectangular rooftop in the Roof abundance.

In conclusion, our methods demonstrate a significant improvement of approximately 50% in unmixing performance as measured by the RMSE of the abundance maps and the SAD of the mixing matrices, requiring only a minimum amount of supervision (e.g., 0.4% labeled pixels). It is important to note that exact abundance maps are not needed during the training process; instead, we leverage the practical and readily obtainable OH labels. Furthermore, while our GRSU methods may exhibit slower computation times due to the iterative nature of our optimization approach, they still maintain a speed advantage over neural network methods, such as MSC and EGU. This not only underlines the efficiency of our methods but also represents a competitive balance between computational speed and unmixing performance.

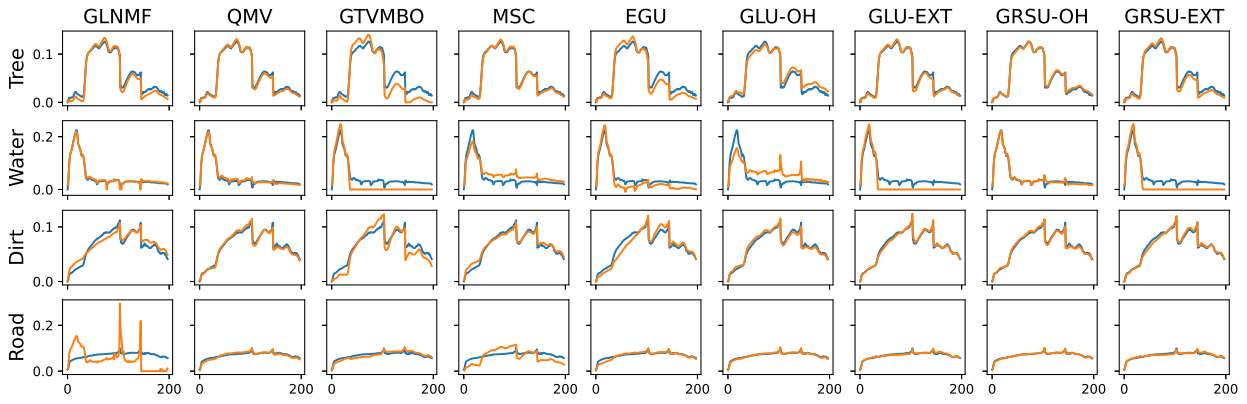
5.3.2 Discussion on the Number of Training Pixels

We discuss the influence of the number of labels sampled by active learning on our semi-supervised methods, GLU-OH, GLU-EXT, GRSU-OH, and GRSU-EXT. The active learning process starts with one random pixel per material as an initial label set, followed by Algorithm 4 (the active learning algorithm) until 5% of the total pixels are reached. We conduct experiments only on the Jasper Ridge and the Apex datasets for demonstration purposes. The corresponding parameters λ, γ, ρ are provided in Table 5.1. The parameter α is designed to balance the numbers of the training pixels and total pixels. Since we have various numbers of training pixels in this experimental setting, we choose $\alpha_0 = 10$ and $\alpha = C_\alpha \alpha_0 / N_{\text{train}}$, where C_α is a constant depending on the dataset and N_{train} is the number of training pixels. In practice, $C_\alpha = 50$ for the Jasper Ridge dataset and $C_\alpha = 10$ for the Apex dataset.

Figure 5.6 shows the changes in the RMSE on abundance and SAD on the endmembers with respect to the number of labeled pixels obtained by active learning. Two abundance RMSE curves in Figure 5.6 (a) and (c) illustrate that our methods trained on one-hot pseudo labels (GLU-OH, GRSU-OH) improve in the very beginning and deteriorate with more labeled pixels (after 1%), which is attributed to the inaccurate information of the OH labels.



(a) Abundance maps: the Jasper Ridge dataset

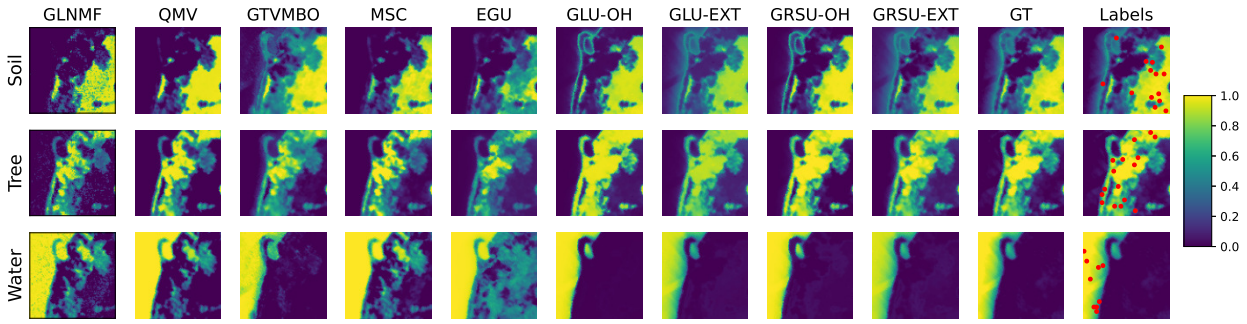


(b) Endmember matrix: the Jasper Ridge dataset

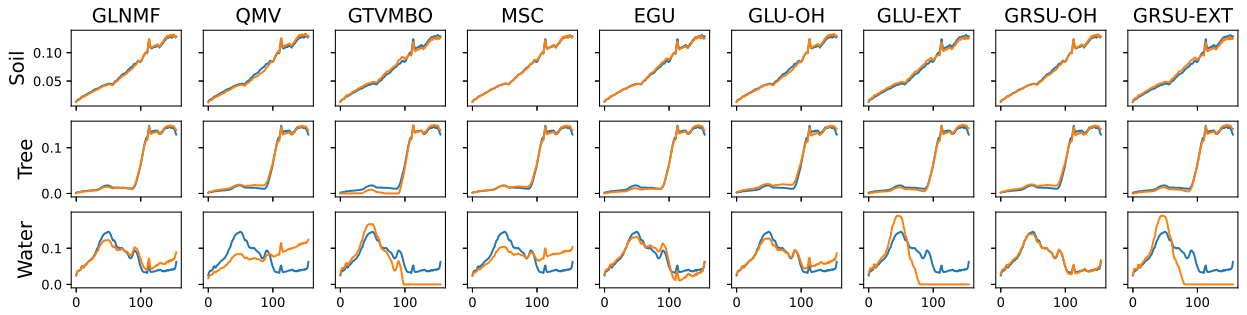
Figure 5.2: Results estimated by different methods on the **Jasper Ridge** dataset. The first five columns are (unsupervised) blind unmixing methods and the following four columns are our semi-supervised methods. Each row corresponds to an endmember, including Tree, Water, Dirt, and Road.

Panel (a) Abundance maps. The last two columns are the ground truth and the label pixels selected by active learning (red dots) for our GLU and GRSU methods. Each red dot corresponds to a labeled pixel, which is enlarged for visual illustration. Note that active learning successfully identifies pixels with high abundance values for Road to acquire labels.

Panel (b) Endmember matrices estimated by different methods (in orange) with the ground truth (in blue). All the endmember vectors are normalized to have the unit norm.



(a) Abundance maps: the Samson dataset

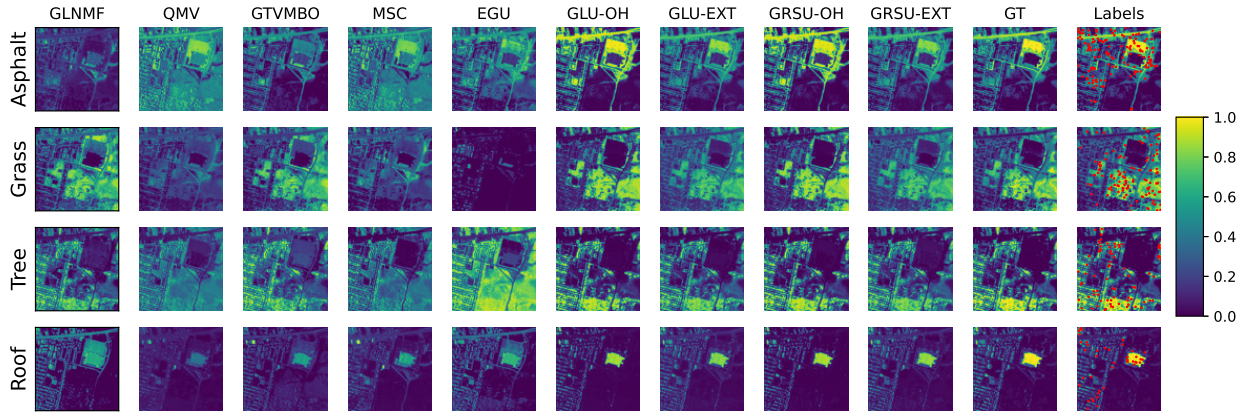


(b) Endmember matrix: the Samson dataset

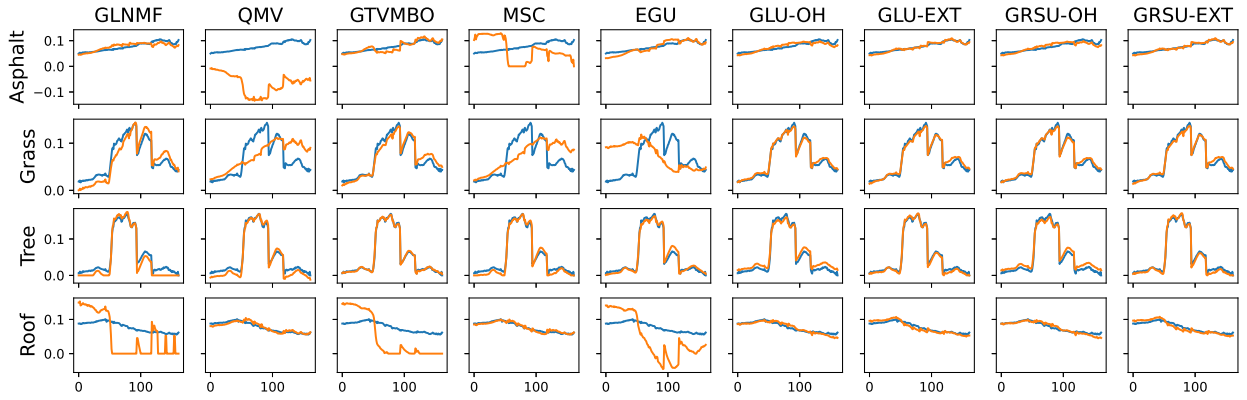
Figure 5.3: Results estimated by different methods on the **Samson** dataset. The first five columns are (unsupervised) blind unmixing methods and the following four columns are our semi-supervised methods. Each row of the plot matrix corresponds to an endmember of the dataset, including Soil, Tree, and Water.

Panel (a) Abundance maps. The last two columns are the ground truth and the label pixels selected by active learning (red dots) for our GLU and GRSU methods. Each red dot corresponds to a labeled pixel, which is enlarged for visual illustration.

Panel (b) Endmember matrices estimated by different methods (in orange) with the ground truth (in blue). All the endmember vectors are normalized to have the unit norm.



(a) Abundance maps: the Urban4 dataset

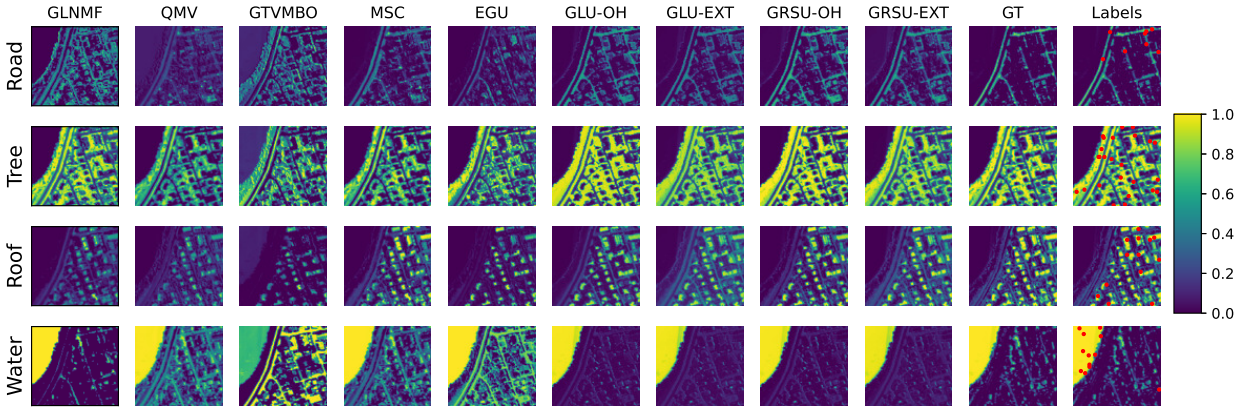


(b) Endmember matrix: the Urban4 dataset

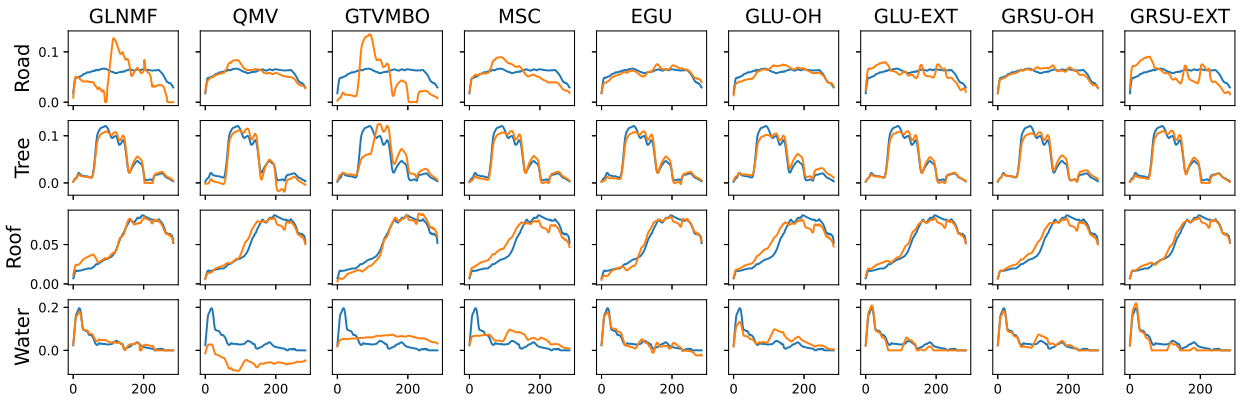
Figure 5.4: Results estimated by different methods on the **Urban4** dataset. The first five columns are (unsupervised) blind unmixing methods and the following four columns are our semi-supervised methods. Each row of the plot matrix corresponds to an endmember of the dataset, including Asphalt, Grass, Tree, and Roof.

Panel (a) Abundance maps. The last two columns are the ground truth and the label pixels selected by active learning (red dots) for our GLU and GRSU methods. Each red dot corresponds to a labeled pixel, which is enlarged for visual illustration. Note that both GLU-OH and GRSU-OH well preserve the contrast of the rectangular rooftop in the Roof abundance.

Panel (b) Endmember matrices estimated by different methods (in orange) with the ground truth (in blue). All the endmember vectors are normalized to have the unit norm.



(a) Abundance maps: the Apex dataset

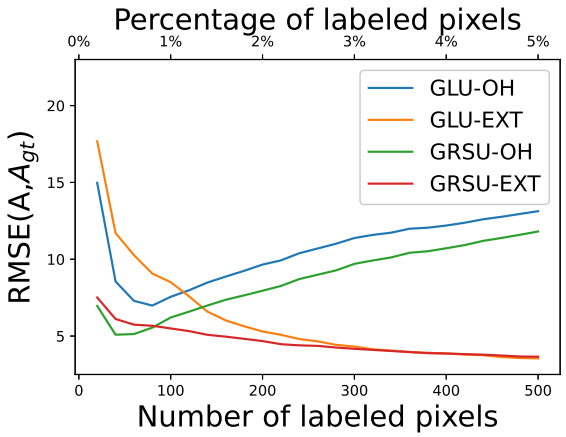


(b) Endmember matrix: the Apex dataset

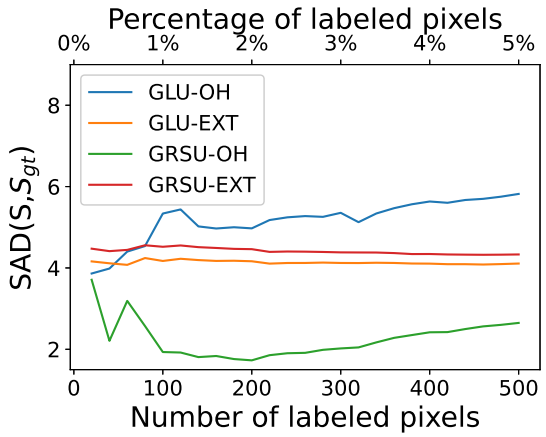
Figure 5.5: Results estimated by different methods on the **Apex** dataset. The first five columns are (unsupervised) blind unmixing methods and the following four columns are our semi-supervised methods. Each row of the plot matrix corresponds to an endmember of the dataset, including Road, Tree, Roof, and Water.

Panel (a) Abundance maps. The last two columns are the ground truth and the label pixels selected by active learning (red dots) for our GLU and GRSU methods. Each red dot corresponds to a labeled pixel, which is enlarged for visual illustration.

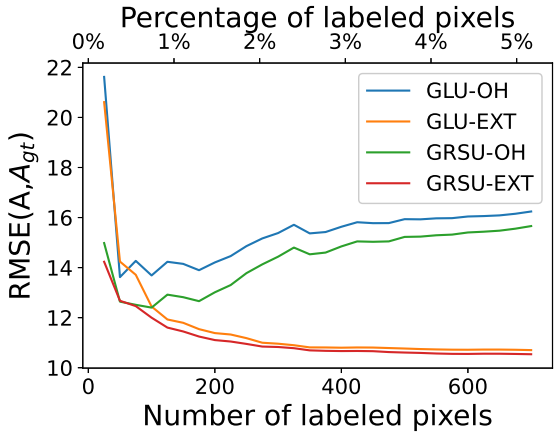
Panel (b) Endmember matrices estimated by different methods (in orange) with the ground truth (in blue). All the endmember vectors are normalized to have the unit norm.



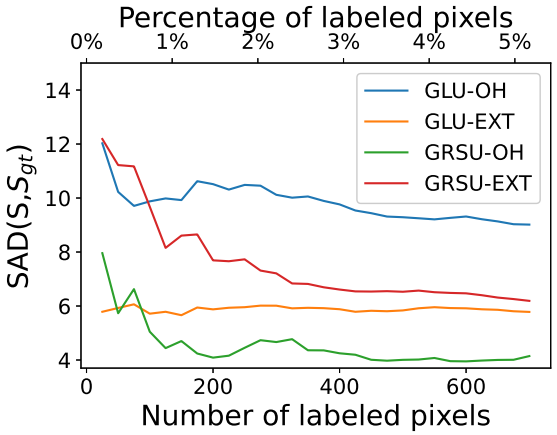
(a) Jasper RMSE



(b) Jasper SAD



(c) Apex RMSE



(d) Apex SAD

Figure 5.6: RMSE and SAD curves with respect to the number of labeled pixels for the Jasper Ridge and the Apex datasets. For each plot, the x-axis on the top shows the percentage of labeled pixels, while the bottom one is the number of labeled pixels. In the active learning process, we apply the MCVOpt and VOpt acquisition functions for the Jasper Ridge and Apex datasets, respectively. Each curve starts with only one random pixel per endmember and samples up to 5% of labeled pixels.

At the beginning of active learning, the most representative pixels are selected, whose ground-truth abundance vectors might be close to a one-hot vector. When we incorporate more OH labels for unmixing, the OH labels are misleading as opposed to the exact abundance map. On the other hand, GLU-EXT and GRSU-EXT always benefit from increasing the number of labeled pixels increases, but with a diminishing gain after around 3% of the label rate. In practice, we would like to use the one-hot pseudo labels for training, since it is much easier to obtain, but not exceeding 1% for the active learning process.

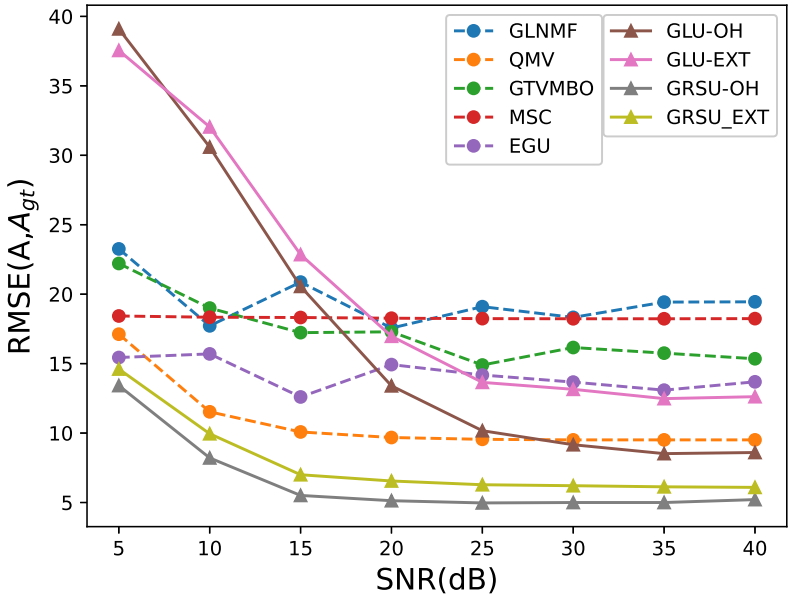
Compared to the RMSE curves, the endmember SAD curves are relatively more stable with respect to the increase in the number of training pixels. Practically, after 2% of the label rate, the SAD values do not change much for all our methods, which implies that only a small number of training pixels are needed to estimate the endmembers.

Overall for both GLU and GRSU, the increase in the training pixels does not always deem improvements in the performances. In fact, including more OH pseudo labels is not beneficial for unmixing. A label rate around 1% would be a good choice experimentally.

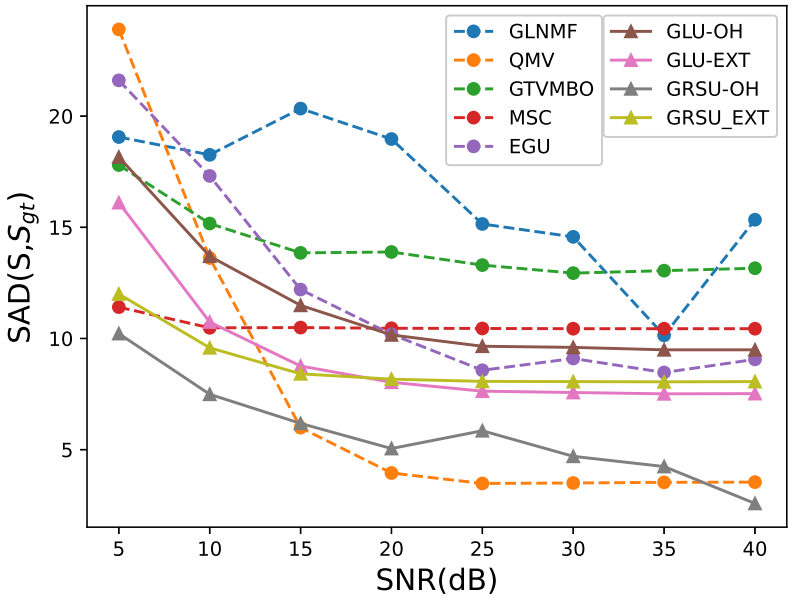
5.3.3 Robustness Study

To provide a quantitative validation of the robustness exhibited by our approach, we have conducted an extensive investigation into the performances of various methods applied to the Jasper Ridge dataset, evaluating them using RMSE (5.21) and SAD (5.22). We add different amounts of Gaussian white noise to the data matrix, resulting in a signal-to-noise ratio (SNR) ranging from 5dB to 40dB with a 5dB increment. It is important to note that a higher SNR value indicates less noisy data.

Figure 5.7 clearly demonstrates that the proposed GRSU method achieves the highest accuracy in estimating the abundance map A, regardless of whether OH or EXT is considered, across all noise levels. When it comes to estimating the spectrum matrix S, GRSU performs best in the low SNR regime and is comparable to QMV when the data is less noisy



(a) RMSE vs SNR



(b) SAD vs SNR

Figure 5.7: RMSE (for A) and SAD (for S) curves concerning the SNR values of noisy input of the Jasper Ridge dataset corrupted by Gaussian white noise. In both panels, the performance of other blind methods is illustrated by dashed lines, whereas solid lines represent our proposed semi-supervised methods.

(SNR>15dB). This result is reasonable, as we only impose two simple constraints on S: nonnegativity and sum-to-one.

In contrast, the GLU method appears to be more sensitive to noise. When the SNR is relatively low, the RMSE and SAD values produced by GLU tend to be large, indicating a higher degree of estimation error. However, it is worth noting that as the SNR gradually increases, there is a sharp decline in both RMSE and SAD values for the GLU method. This suggests that while GLU may struggle in highly noisy environments, its performance improves significantly as the data becomes cleaner.

5.4 Conclusion

In this chapter, we propose two semi-supervised hyperspectral models, graph learning unmixing (GLU) and graph-regularized semi-supervised unmixing (GRSU). GLU applies the graph Laplace learning directly to solve the HSU problem by regarding the class probability as the abundance map. Initialized by GLU, GRSU estimates the abundance map A and endmember spectrum matrix S by solving an energy-minimizing problem via an iterative ADMM scheme. We extended the graph Laplace learning to a regularized version and explored the close-form solutions for efficient computation. This regularized graph Laplace learning is a subproblem in the ADMM iteration process. We conducted extensive experiments using four standard hyperspectral datasets to compare our semi-supervised methods with five state-of-the-art methods in hyperspectral blind unmixing. All the results demonstrated the proposed GLU and GRSU methods have a significant improvement with a minimum amount of supervision. Furthermore, both methods can take either the ground-truth abundance maps or the one-hot pseudo labels as the training information, the latter of which is much easier to obtain since it only requires determining the major endmember of each training pixel. According to our experiments, it is unnecessary to require the ground-truth abundance maps to feed in the semi-supervised framework. Sometimes, models trained on pseudo labels yield better

performance than the exact values. We also discussed the influence of the number of training pixels on the model performance, revealing that a label rate of 1% is experimentally sufficient for satisfactory results. In addition, our GRSU method is more robust than Gaussian white noise. Our new methods have great potential for real-world problems since they do not need a ground truth abundance map and can work with pseudo-labels instead.

There are several promising directions following this work. First, scaling factor [DVH16] and spectral variability [HYC19b] terms can be incorporated into our model to further enhance the unmixing performance. Second, Figure 5.7 (b) suggests a need for regularizations of the spectral matrix, in addition to nonnegativity and sum-to-one constraints, to improve the robustness of the proposed methods. Lastly, the current graph Laplacian learning solver can be replaced with neural networks or graph neural network techniques.

CHAPTER 6

AutoKG: Efficient Automated Knowledge Graph Generation for Language Models

This chapter reuses IEEE copyrighted material from the author's publications [CB23] with the approval of the senior author Andrea L. Bertozzi and following the requirements outlined by IEEE for thesis/dissertation reuse.[1]

As introduced in previous chapters, graph-based methods have found extensive applications in image analysis. These methods leverage graphs as a mathematical tool to model the spatial relationships among pixels, regions, and objects in images, enabling a deeper understanding and analysis of image content. Although this thesis focuses on graph-based approaches in image analysis, we note that integrating knowledge bases with large language models (LLMs) has become a research hotspot in natural language processing. Traditional methods for combining the two rely on semantic similarity search, where the query and text blocks in the knowledge base are vectorized, and relevant knowledge is retrieved based on the similarity in the vector space. However, these methods struggle to capture the complex associations between pieces of knowledge. We propose AutoKG, a lightweight and efficient approach for automated knowledge graph (KG) construction to address this limitation. AutoKG first extracts keywords from the text using a language model and then computes the association weights between keywords using graph Laplacian learning, automatically constructing a knowledge graph. We employ a hybrid search scheme that combines vector similarity and graph-based associations during knowledge retrieval. Preliminary ex-

[1] ©2023 IEEE. Reprinted, with permission, from [CB23]

periments demonstrate that AutoKG can capture the connections between knowledge more comprehensively than pure semantic similarity search, generating richer and more coherent language model outputs.

This chapter demonstrates that traditional graph Laplacian-based methods can be applied not only to image analysis but also to current research hotspots such as large language models. It inspires us to explore the application of graph-based methods in various domains and tasks of machine learning and artificial intelligence.

6.1 Introduction

Large Language Models (LLMs) such as BERT [DCL18], RoBERTA [LOG19], T5 [RSR20], and PaLM [CND23], are intricately designed architectures equipped with an extensive number of parameters. These models have been rigorously pre-trained on vast and diverse corpora, thereby enabling them to excel in a wide array of Natural Language Processing (NLP) tasks, from language understanding to both conditional and unconditional text generation [TMZ22, ZYL22]. These advancements have been heralded as a step toward higher-bandwidth human-computer interactions. However, their deployment faces significant challenges. On one hand, LLMs exhibit a tendency for "hallucinations" [WKR20, JLF23], providing plausible yet nonfactual predictions. On the other hand, the black-box nature of LLMs compromises both interpretability and factual accuracy, often resulting in erroneous statements despite memorizing facts during training [PRL19, SCM22].

Knowledge in natural language can be externally sourced from a retrievable database, reducing hallucinations and enhancing the interpretability of LLMs [MDL23]. Utilizing dense neural retrievers, which employ dense query and document vectors generated by a neural network [AYK21], the system can evaluate the semantic similarity to an information-seeking query by calculating the embedding vector similarity across related concepts [LPP20, LET21].

To go beyond mere semantic similarity in information retrieval and augment the reasoning capabilities of LLMs, two advanced methodologies are particularly transformative: prompt engineering like the Chain-of-thought prompting, and the incorporation of Knowledge Graphs (KGs) [PLW23]. The former, chain-of-thought prompting, provides a framework for advanced reasoning by generating paths of explanations and predictions that are cross-verified through knowledge retrieval [HZR22, TBK22]. While this method offers significant benefits, it is not the primary focus of the study in this chapter. As for the latter, KGs offer LLMs a structured and efficient way to address their limitations in factual accuracy and reasoning [XYC19, PLW23]. KGs not only provide accurate and explicit knowledge crucial for various applications [JPC21] but are also known for their symbolic reasoning capabilities to produce interpretable results [ZCZ21]. These graphs are dynamic, continuously evolving with the addition of new knowledge [MCH18], and can be specialized for domain-specific requirements [Abu21].

In this chapter, our emphasis is on techniques of automated KG generation and incorporation with LLMs. Most of the works related to these two tasks rely intensively on the ongoing training of neural networks [PLW23, ZWL23], which is both difficult to employ and less flexible for on-the-fly updates. Traditional KG construction approach uses NLP techniques for entity recognition [NS07, GS96], or keyword identification based on term frequency [SB88, Ram03], followed by determining relationship strength through word proximity [MBS09]. Current automated techniques necessitate neural network training [LLH23, WPG21, WLR23]. As for the interaction between KGs and LLMs, neural networks are trained to let LLMs understand the information retrieved from KGs [TSW23, YBR22].

The recent advancements in LLMs make us think much more simply about the automatic generation of KGs and the integration of LLMs with KGs. State-of-the-art LLMs such as ChatGPT[2], BARD[3], and LLAMA [TMS23] have demonstrated impressive reasoning

[2] `https://openai.com/blog/chatgpt`

[3] `https://blog.google/technology/ai/bard-google-ai-search-updates/`

capabilities [BCL23, ASG23]. Given sufficient information, they can independently execute effective inference. This observation suggests an opportunity to simplify the KG structure: perhaps the intricate relational patterns found in traditional KGs could be simplified into basic strength indicators of association. Consequently, specific relationships are implicitly conveyed to the model through corpus blocks associated with the KG. In addition, we can provide retrieved keywords and the related corpus directly in the prompt rather than training a network to let LLMs understand the retrieved subgraph structure.

Motivated by these ideas, we make the following contributions in this chapter:

1. We introduce AutoKG, an innovative method for automated KG generation, based on a knowledge base comprised of text blocks. AutoKG circumvents the need for training or fine-tuning neural networks, employs pretrained LLMs for extracting keywords as nodes, and applies graph Laplace learning to evaluate the edge weights between these keywords. The output is a simplified KG, where edges lack attributes and directionality, possessing only a weight that signifies the relevance between nodes.

2. We present a hybrid search strategy in tandem with prompt engineering, which empowers large LLMs to effectively utilize information from the generated KGs. This approach simultaneously searches for semantically relevant corpora based on embedding vectors and the most pertinent adjacent information within the knowledge graphs.

The KG constructed here is a simplified version compared to traditional KGs, which are typically composed of relations in the form of triplets. Firstly, nodes in AutoKG are not entities in the usual sense; they are more abstract keywords. These keywords can represent entities, concepts, or any content that serves as a foundation for search. Additionally, instead of directed edges with specific semantic meanings found in traditional KGs, AutoKG utilizes undirected edges with a single weight value. The node keywords are extracted from the knowledge base with the aid of LLMs, while the graph structure is algorithmically derived. Such a KG can be efficiently stored with just a keyword list and a sparse adjacency matrix.

Section 6.2 explains the detailed process of automated KG generation, while Section 6.3 describes the hybrid search method. An essential highlight is that our proposed techniques require no neural network training or fine-tuning.

6.2 Automated KG Generation

In this section, we introduce our proposed approach, AutoKG, for automated KG generation. The training aspects of the LLM are not the focus of this article. We operate under the assumption that the LLM is already pre-trained and is accompanied by a corresponding vector embedding model. Specifically, we have employed OpenAI's *gpt-4* or *gpt-3.5-turbo-16k* as the LLM and the *text-embedding-ada-002* as the embedding model.

Consider a scenario involving an external knowledge base, comprised of discrete text blocks. AutoKG constructs a KG where the nodes represent keywords extracted from the external knowledge base. The edges between these nodes carry a single non-negative integer weight, signifying the strength of the association between the connected keywords. AutoKG encompasses two primary steps: the extraction of keywords, which correspond to the nodes in the graph, and the establishment of relationships between these keywords, represented by the edges in the graph. It is worth noting that the pretrained LLM is employed only in the keyword extraction step of the process. Figure 6.1 is the flowchart of the KG construction.

6.2.1 Keywords Extraction

Let the external knowledge base be denoted by $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where each $\mathbf{x}_i$ is a block of text with the maximum length of T tokens, represented as a string. The corresponding embedding vectors for these text blocks are encapsulated in $\mathcal{V} = \{\mathbf{v}(\mathbf{x}_1), \mathbf{v}(\mathbf{x}_2), \ldots, \mathbf{v}(\mathbf{x}_N)\} \subset \mathbb{R}^d$, where $\mathbf{v}$ is the embedding projection from string to $\mathbb{R}^d$. We extract keywords from the knowledge base X with unsupervised clustering algorithms and the assistance of LLMs.

Algorithm 7 outlines the keyword extraction process. The algorithm takes as input all

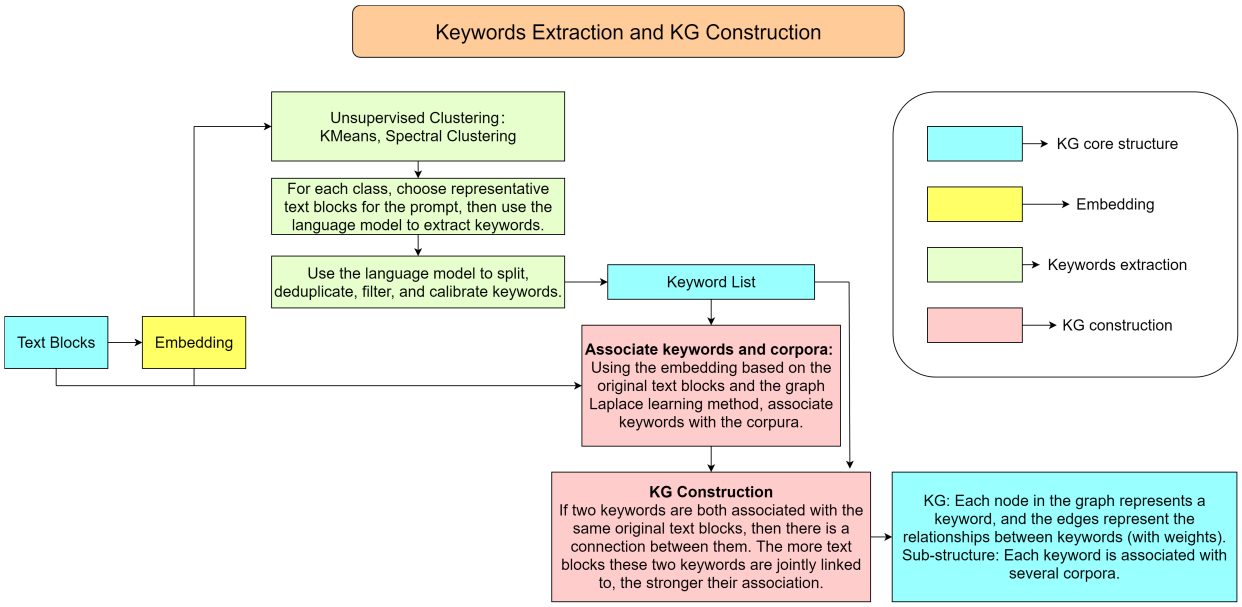


Figure 6.1: Flowchart of the KG Construction Process. This figure illustrates the different steps involved in the construction of the KG. The blue blocks represent the core components of the KG, the yellow blocks indicate the embedding process, the green blocks focus on keyword extraction, and the red blocks correspond to the establishment of relationships between keywords and the corpus as well as among the keywords themselves.

text blocks and their corresponding embedding vectors X and $\mathcal{V}$, along with pre-defined parameters: n for the number of clusters, c for the number of text blocks to select, and l_1, l_2 as keyword extraction parameters. Additionally, the algorithm also utilizes a parameter m to specify the number of sampled previous keywords. Two unsupervised clustering algorithms, K-means clustering [Mac67, Llo82] and spectral clustering [Von07], are applied to cluster the knowledge base. For each cluster identified, we sample $2c$ text blocks, with c closest to the cluster center and c randomly selected, to capture both the global and centered information. The LLM is used twice in this algorithm. First, it extracts keywords from a selection of $2c$ text blocks, guided by the parameters l_1 and l_2, while avoiding the sampled m previous keywords. Second, the same LLM is employed to filter and refine the extracted keywords.

The construction of the prompts for these applications strictly follows the format outlined in Table 6.1. A specific prompt example for the keyword extraction is given in the Appendix. Specifically, each prompt is formed by concatenating the *Task Information*, *Input Information*, *Additional Requirements*, and *Outputs*. It is essential to note that within each task, the length of the prompt sections corresponding to *Task Information* and *Additional Requirements* is fixed.

For Task 1, which deals with keyword extraction, the maximum input length is set to $2cT + m(l_2 + 1)$, where T represents the token length of a single text block. Note that each keyword can have a length of up to $l_2 + 1$ tokens when accounting for potential separators such as commas. Similarly, the maximum output length is $l_1(l_2+1)$, where l_1 is the maximum number of keywords and l_2 is the maximum token length of each keyword. Since Task 1 is applied once for each of the n clusters generated by the two clustering methods, the total maximum token usage for Task 1 would be $2n(2cT + (m + l_1)(l_2 + 1))$. This process yields a maximum of $2nl_1$ extracted keywords. For Task 2, which involves filtering and refining the keywords, the maximum lengths for both the input and output are governed by the formula $2nl_1(l_2 + 1)$. In summary, the maximum usage of tokens $M_{\text{tokens KG}}$ for the keyword extraction process is

$$M_{\text{tokens KG}} = 2n(2cT + (m + 2l_1)(l_2 + 1)) + L_F, \tag{6.1}$$

where L_F is the fixed total length of tokens of the task information and additional requirement parts.

6.2.2 Graph Structure Construction

In this section, we detail how to construct a KG based on the keywords extracted in Section 6.2.1. Specifically, we establish whether there are edges between keywords and how to weight these edges. We propose a method based on label propagation on the graph, a step that does not require the involvement of any LLM.

Algorithm 7 Algorithm for Keyword Extraction in AutoKG

Require: All text blocks and their corresponding embedding vectors X and $\mathcal{V}$, pre-defined parameters n (number of clusters), c (number of text blocks to select), l_1, l_2 (keyword extraction parameters), m (number of sampled previous keywords)

Ensure: A set of extracted keywords $\mathcal{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M\}$

$\mathcal{K} = \emptyset$
for each clustering algorithm P in {k-means, spectral clustering} **do**
  Cluster $\mathcal{V}$ into n clusters $\mathcal{V}_i^P,\ i = 1, 2, \ldots, n$ using algorithm P
  for i = 1,2,. . . ,n **do**
    Randomly select c text blocks and c nearest to the cluster center from cluster $\mathcal{V}_i^P$
    if $|\mathcal{K}| > m$ **then**
      Select a subset $\mathcal{K}_s \subset \mathcal{K}$ such that $|\mathcal{K}_s| = m$
    else
      $\mathcal{K}_s = \mathcal{K}$
    end if
    Include these $2c$ text blocks and previous keywords $\mathcal{K}_s$ in a prompt for keyword extraction
    Use LLM, extract up to l_1 keywords of maximum token length l_2, collected as $\mathcal{K}_i^P$
    Update $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_i^P$
  end for
end for
Filter and refine $\mathcal{K}$ using a LLM to obtain the final keyword list
return $\mathcal{K}$

ID	***Task***	***Input Information***	***Additional Requirements***	***Outputs***
1	Keywords Extraction	1.Sampled text blocks 2.Previous keywords	1. Avoid previous keywords 2. Output up to l_1 keywords 3. Each keyword is at most l_2 tokens	Extracted Keywords
2	Refining Keywords	Original Keywords	Concentration, deduplication splitting, deletion	Refined Keywords
3	Response to the query	1. Original query 2. Related text blocks 3. Related keywords	Indicate the hybrid search approach: Direct, via keywords, or KG adjacency search	Final Response

Table 6.1: Prompt Construction for Different Tasks Using LLM

We firstly construct a similarity graph $G^{\text{text}} = (X, W^{\text{text}})$ where X is the set of text blocks serving as the nodes, and W^{text} is the weight matrix for the edges, according to Secion 2.2.1. We use the embedding vectors $\mathbf{v}(\mathbf{x}_i), \mathbf{v}(\mathbf{x}_j)$ to calculate the similarity weight between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Then, we utilize the graph $G^{\text{text}} = (X, W^{\text{text}})$, constructed on text blocks, to establish a keyword-based KG $G^{\text{keyword}} = (\mathcal{K}, W^{\text{keyword}})$. Here, $\mathcal{K}$ is the set of keywords, and W^{keyword} is the weight matrix for the edges. In this matrix, W_{ij}^{keyword} quantifies the strength of association between keywords $\mathbf{k}_i$ and $\mathbf{k}_j$. Importantly, this association is not semantic but is reflected across the entire corpus in the knowledge base. Specifically, W_{ij}^{keyword} corresponds to the count of text blocks that are simultaneously associated with both keywords $\mathbf{k}_i$ and $\mathbf{k}_j$.

Algorithm 8 establishes the relationship between a keyword and text blocks. The core idea is to select a subset of text blocks that are closest to the keyword as positive data, and another subset that is farthest as negative data. We then employ graph Laplace learning [ZGL03] based on the graph structure $G^{\text{text}} = (X, W^{\text{text}})$ that we have previously constructed for text blocks. The graph Laplace learning is a semi-supervised learning method on graphs,

Algorithm 8 Identifying Keyword to Text Block Association

Require: Keyword $\mathbf{k}$, Set of text blocks X, Forward relation parameter n_1, Backward relation parameter n_2

Ensure: $X^{\mathbf{k}} \subset X$, the subset of X associated with $\mathbf{k}$

1: Obtain the embedding vector $\mathbf{v}(\mathbf{k})$.

2: Find the n_1 nearest vectors in X to $\mathbf{v}(\mathbf{k})$ (label them as 1) and n_2 farthest vectors (label them as 0).

3: In the text-block graph $G^{\text{text}} = (X, W^{\text{text}})$, use the graph Laplace learning algorithm [ZGL03] to label the remaining nodes based on these $n_1 + n_2$ labeled nodes. Obtain a real-valued function $\mathbf{u} : X \to [0, 1]$ on the graph nodes.

4: Define $X^{\mathbf{k}} = \{\mathbf{x}_i \in X : \mathbf{u}(\mathbf{x}_i) \geq 0.5\}$

5: **return** $X^{\mathbf{k}}$

utilizing the harmonic property of the solution function $\mathbf{u} : X \to [0, 1]$ to diffuse the label values from a subset of labeled nodes to other unlabeled nodes in the graph. The text blocks that are classified towards the positive side (with a node function value $u \geq 0.5$) are considered to be associated with the keyword.

The association weight W_{ij}^{keyword} between $\mathbf{k}_i$ and $\mathbf{k}_j$ is defined as follows:

$$W_{ij}^{\text{keyword}} = W_{ji}^{\text{keyword}} = |X^{\mathbf{k}_i} \cap X^{\mathbf{k}_j}|, \tag{6.2}$$

where $X^{\mathbf{k}} = \{\mathbf{x}_i \in X : \mathbf{u}(\mathbf{x}_i) \geq 0.5\}$. With this, we complete the construction of the keyword-based KG G^{keyword}, which is built upon the text block graph G^{text}.

6.2.3 Time Complexity Analyzation

This section analyzes the efficiency of the AutoKG method. The token consumption required for KG construction in the AutoKG method has the upper bound according to Eq. 6.1. The efficiency of the algorithm is mainly influenced by three aspects:

1. **Constructing the similarity graph based on text blocks** $G^{\text{text}} = (X, W^{\text{text}})$: An

approximate nearest neighbor search [AMN98] is employed for KNN search, leading to a complexity of $\mathcal{O}N \log N)$.

2. **Clustering algorithm**: Since both K-means clustering [Mac67, Llo82] and spectral clustering [Von07] are NP-hard, we bound the complexity by $I_{\max}$, the preset maximum number of iterations. Spectral clustering is essentially the Kmeans method augmented with an eigen-decomposition of the graph Laplacian. The time complexity here is mainly dominated by the Kmeans method and is $\mathcal{O}NndI_{\max})$, where n is the number of clusters, and d is the vector dimension (1536 for OpenAI's embedding model).

3. **Graph Laplace learning**: Given that our graph Laplacian matrix is sparse, employing the conjugate gradient method to solve the graph Laplace learning problem results in a time complexity of $\mathcal{O}\hat{N}\sqrt{\kappa})$, where $\hat{N}$ represents the count of non-zero elements in the graph Laplacian matrix, and $\sqrt{\kappa}$ denotes the condition number. We have the upper bound for $\hat{N}$ as $2KN$, where K is the number of nearest neighbors.

Considering these factors, for large N and if preconditioning techniques can keep the condition number of the graph Laplacian matrix small, our automated KG construction algorithm should operate with a time complexity of

$$\mathcal{O}N \log N + NndI_{\max} + 2KN\sqrt{\kappa}) = \mathcal{O}N \log N + Nn),$$

where the number of clusters practically depends on N.

6.2.4 Remarks

In the process of generating the entire KG, there are several points to be considered:

- Although the keywords are extracted from clusters of text blocks, we do not take into account the previous clustering results when establishing the relationship between keywords and text blocks. This is because the same keyword may be included in multiple clusters.

- When constructing the relationship between keywords, we did not incorporate the embedding vectors of the keywords into the graph for the graph Laplace learning process. There are two reasons for this decision: first, we do not need to update the graph structure when selecting different keywords; second, empirically speaking, the embedding vectors of the keywords tend to be quite distant from the embedding vectors of the text blocks. Therefore, including them in the initial label data for Laplace learning might be meaningless.

Our approach considerably outperforms these conventional methods in both keyword extraction and relationship construction. The primary shortcoming of traditional techniques is their reliance on a fixed set of words, leading to a significant loss of related information and often producing overly localized insights. In terms of keyword extraction, our method leverages the capabilities of LLMs, allowing for the refining of keywords that are more central to the topic at hand, rather than merely being high-frequency terms. When it comes to relationship construction, our strategy is grounded in a macroscopic algorithm on graphs of all text blocks. This approach encompasses the information from the entire knowledge base of text blocks, providing a more comprehensive perspective compared to relationships derived from local distances.

6.3 hybrid search: Incorporating KG and LLM

In this section, we propose a hybrid search approach, based on the KG generated according to Section 6.2. For a given query, the search results using this hybrid search strategy include not only the text blocks that are semantically related to the query but also additional associative information sourced from the KG. This supplementary data serves to provide more detailed and in-depth reasoning for further analysis by the model. The incorporation of a KG allows us to capture complex relationships between different entities, thereby enriching the contextual understanding of the query.

Algorithm 9 hybrid search Algorithm

Require: Query $\mathbf{q}$, embedding vector $\mathbf{v}(\mathbf{q})$, Parameters $(s_0^t, s_1^k, s_1^t, s_2^k, s_2^t)$

Ensure: Set X_{final} containing text blocks related to $\mathbf{q}$, and Set $\mathcal{K}_{\text{final}}$ containing keywords related to $\mathbf{q}$

Step 1: Vector Similarity Search

Find the closest s_0^t text blocks in X to $\mathbf{v}(\mathbf{q})$

$X_0 \leftarrow$ set of closest s_0^t text blocks

Step 2: Similar Keyword Search

Find the closest s_1^k keywords in $\mathcal{K}$ to $\mathbf{v}(\mathbf{q})$

$\mathcal{K}_1 \leftarrow$ set of closest s_1^k keywords

For each $\mathbf{k}$ in $\mathcal{K}_1$, find the closest s_1^t text blocks in X

$X_1 \leftarrow$ merged set of closest s_1^t text blocks for each k in $\mathcal{K}_1$

Step 3: Keyword Adjacency Search

For each $\mathbf{k}$ in $\mathcal{K}_1$, find s_2^k strongest connected keywords according to W^{keyword}

$\mathcal{K}_2 \leftarrow$ merged set of s_2^k strongest connected keywords for each k in $\mathcal{K}_1$

For each $\mathbf{k}$ in $\mathcal{K}_2$, find the closest s_2^t text blocks in X

$X_2 \leftarrow$ merged set of closest s_2^t text blocks for each k in $\mathcal{K}_2$

$X_{\text{final}} \leftarrow X_0 \cup X_1 \cup X_2$

$\mathcal{K}_{\text{final}} \leftarrow \mathcal{K}_1 \cup \mathcal{K}_2$

return $X_{\text{final}}, \mathcal{K}_{\text{final}}$

In our proposed hybrid search approach, we have devised a multi-stage search process that incorporates both direct text block search and keyword-based searching guided by the KG. This process is detailed in Algorithm 9. Initially, we perform the initial search by computing the text blocks that are closest to the given query embedding vector. Then, we turn to the KG and identify the keywords that are closest to the query, along with text blocks associated with these keywords. Lastly, we identify additional keywords that have the strongest association with the previously identified ones, based on the weight matrix in

the KG, and accordingly search for related text blocks. The algorithm returns not just a set of text blocks that are highly relevant to the query but also a set of keywords that are closely connected to the query.

To estimate the maximum number of tokens returned by the hybrid search, we consider the maximum number of tokens T for a single text block and l_2 for a single keyword. The total number of keywords retrieved will be $s_1^k + s_1^k \cdot s_2^k$, and the total number of text blocks will be $s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t$. Therefore, the maximum number of tokens $M_{\text{tokens QA}}$ can be calculated as:

$$M_{\text{tokens QA}} = s_1^k \cdot l_2 \cdot (1 + s_2^k) + T \cdot (s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t). \tag{6.3}$$

In practical applications, the actual number of tokens obtained through the search often falls below the theoretical maximum. This is because there is substantial overlap between the text blocks and keywords discovered via different search methods. Subsequently, the retrieved information is incorporated into the prompt to enhance the LLM's response to the original query. For details on prompt construction, one may refer to Task 3 in Table 6.1. A specific prompt example is provided in the Appendix. Importantly, an adaptive approach can be employed during the prompt construction to ensure that the maximum token limit for the LLM is not exceeded. Text blocks can be added sequentially until the token limit is reached.

6.4 Experiments and Results

In this section, our primary goal is to demonstrate through experiments that our proposed AutoKG approach provides significantly better responses while maintaining a comparable efficiency, compared with the retrieval-augmented generation (RAG) method based on semantic vector similarity [LPP20, LET21]. Our approach that combines AutoKG and hybrid search extracts more valuable information for the model than RAG which relies on semantic vector similarity search.

Unfortunately, we encountered challenges in identifying a suitable dataset to conduct these experiments. We attempted to utilize the WikiWhy dataset [HSC23], which is designed to evaluate the reasoning capability of models. The dataset comprises approximately 9,000 entries. Each entry contains a paragraph of content, spanning between 100 to 200 words. Based on this content, every entry provides a "why" question along with its corresponding cause-effect relationship and explanation. When we employ the hybrid search based on AutoKG or the semantic vector similarity search of RAG, we can easily retrieve the content corresponding to the given question and instruct the model to answer based on that content. In both methods, the model's responses are almost identical. Since the 9,000 entries are relatively independent of each other, cross-entry data retrieval provided by our method doesn't significantly contribute to answering the questions.

As a consequence, we adopt qualitative approaches rather than employing numerical metrics to evaluate the experimental performance of our method. First, we provide a simple example to explain why our AutoKG with hybrid search approach has benefits compared to methods based on semantic vector similarity search. Next, we present a detailed example based on all 40 references of this article and the associated subgraph from the KG used during the query. Finally, we compare the efficiency of hybrid search and semantic vector similarity search from both theoretical and experimental perspectives.

6.4.1 A Simple Example: Why We Need KG?

Consider a simple knowledge base that contains text blocks detailing a day in the life of an individual named Alex, along with related information. The core narrative is that after leaving his home in the morning, Alex goes to Cafe A to buy a coffee and then takes a bus to Company B for work. Interspersed within the knowledge base are numerous pieces of granular information such as conversations Alex had with the barista at the cafe, the coffee order details, dialogues on the bus, as well as conversations at his workplace, and so forth.

The point of interest here is how a model would answer the question: *"Was it raining this*

morning when Alex left his home?" under the assumption that there is no direct answer to this question and no content about the weather in the knowledge base. We aim to compare the responses given the support information retrieved using our method versus that retrieved through semantic similarity search. Within the knowledge base, there are two indirect pieces of information hinting at the weather conditions:

1. Related to Cafe A: "Many people were chatting and drinking coffee in the square outside Cafe A."

2. Related to Company B: "The car wash located downstairs of Company B was bustling with business today."

Both these snippets subtly suggest that it was not raining.

Given that the question is primarily about Alex and the weather, the information retrieved from the knowledge base through semantic similarity vector search would only be about Alex (as there is no direct information about the weather). The search results would primarily outline his movements throughout the day. Even with an increase in search entries, it would mostly retrieve additional miscellaneous details, like his coffee order and dialogues. Unfortunately, these details do not contain any hints to infer the day's weather.

On the other hand, employing AutoKG with a hybrid search approach yields different results. During the KG generation process, we extract keywords such as Alex, Cafe A, and Company B. With the hybrid search, the initial step uses the input question to retrieve the keyword Alex. Then, the adjacency search identifies Cafe A and Company B as related keywords. Subsequently, text blocks are sought based on these keywords, resulting in the identification of implicit weather-related information. This example illustrates the utility of the hybrid search. Semantic similarity alone can lack cross-topic connections. It tends to retrieve many minor details within the scope of a given question. When searching with the KG constructed using the AutoKG method, the breadth and diversity of the retrieved

information is enhanced. Moreover, prior work has easily substantiated GPT-4's capability to reason effectively with provided clues [BCL23, ASG23].

From the dialogue record with GPT-4 in the Appendix, it is evident that GPT-4 can accurately infer that it did not rain today when given clues about today's weather. However, when only provided with information about Alex from the semantic similarity vector search, it cannot make any predictions about today's weather.

6.4.2 An Example with Article References

We present a concrete example utilizing content from the 42 references cited in this chapter. The resulting KG is interactively queried using the hybrid search method outlined above. Both the KG generation and subsequent querying processes were performed using the *gpt-3.5-turbo-16k* model, chosen to minimize cost. The 40 references, once segmented, comprise 5,261 text blocks, each less than 201 tokens in length. For the keyword extraction process, as per Algorithm 7, the parameters are: $n = 15, c = 15, l_1 = 10, l_2 = 3, m = 300$. For Algorithm 8, we use the parameters $n_1 = 5$ and $n_2 = 35$. The entire KG construction consumes 137,516 tokens, which is less than the theoretical maximum of 181,280 tokens given by Eq. 6.1. This calculation of the theoretical maximum does not account for the fixed total length of tokens pertaining to task information and additional requirement parts.

The constructed KG comprises 461 nodes (extracted keywords) with its adjacency matrix containing 40,458 non-zero elements. The node with the highest degree in the graph is connected to 289 neighbors. There are 353 nodes whose degree is less than 92, which is 20% of the maximum possible degree of 460. The entire process of KG construction took approximately ten minutes. All computations, excluding calls to the OpenAI API, are carried out on a CPU with an Intel i9-9900. Both keyword extraction and KG construction take approximately five minutes each. For the subsequent hybrid search described in Algorithm 9, we use the parameters $(s_0^t = 15, s_1^k = 5, s_1^t = 3, s_2^k = 3, s_2^t = 2)$ and ensure, through an adaptive approach, that the input prompt remains under 10,000 tokens in length. The

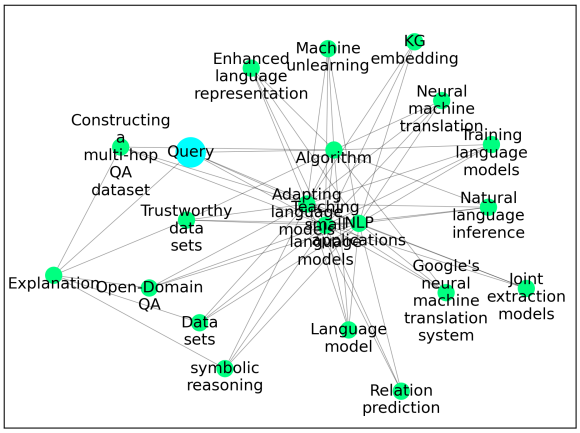


Figure 6.2: Subgraph Visualization: Keyword Nodes

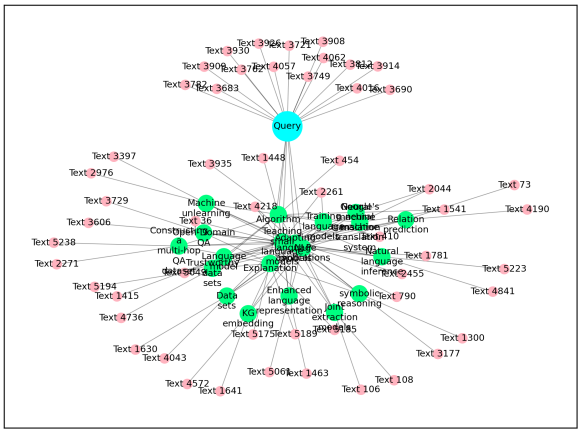


Figure 6.3: Subgraph Visualization: Keyword and Text Block Nodes

maximum length of response is set as 1024. As an illustrative example, when querying: *"Please introduce PaLM in detail, and tell me about related applications."*, the temporary KG structure during the hybrid search is shown in the Figures 6.2 and 6.3. Both images represent subgraphs of the same KG, with the input query depicted in blue nodes. The image on the left (Figure 6.2) showcases only the keyword nodes (in green), while the image on the right (Figure 6.3) includes the additionally retrieved text blocks (in pink nodes). The edges displayed are those connecting similar keywords directly retrieved from the query (shown as inner circle nodes in the left figure) as well as edges connecting these similar keywords to the keywords obtained via adjacency search (connecting the inner and outer circles in the left figure). While there may be existing edges between the outer circle keywords, they are omitted from the visualization for clarity. The model has a lengthy response which is shown in the Appendix. Please check our GitHub repository [4] for those interested in further exploration.

[4] `https://github.com/wispcarey/AutoKG`

6.4.3 Efficiency Analyzation

Given the flexibility in regulating the volume of retrieved information, both the proposed method and the RAG approach can, in theory, support knowledge bases of any size. This means they can encompass any number of text blocks, each subject to the maximum token limit. As outlined in Section 6.2.3, the efficiency of the AutoKG method for automated knowledge graph construction is $\mathcal{O}N \log N)$ when the number of text blocks N is large.

The constructed keyword KG contains M keywords where $M < N$ (empirically, $M \approx 0.1N$). During the hybrid search process, with parameters $(s_0^t, s_1^k, s_1^t, s_2^k, s_2^t)$, the overall time complexity for the search is:

$$\mathcal{O}(s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t)N) + \mathcal{O}(s_1^k + s_1^k \cdot s_2^k)M). \tag{6.4}$$

For the semantic vector similarity search method to retrieve the same volume of text blocks, the time complexity is:

$$\mathcal{O}(s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t)N). \tag{6.5}$$

From the above, it's evident that the time complexity of our hybrid search approach is the same as that of the semantic vector similarity search. For sufficiently large N, both complexities tend towards $\mathcal{O}N)$.

Based on the KG generated from the 40 references of this article, as described in Section 6.4.2, we perform a hybrid search using parameters $(s_0^t = 15, s_1^k = 5, s_1^t = 3, s_2^k = 3, s_2^t = 2)$. The theoretical maximum number of text blocks that can be searched using this configuration is 60. For comparison, we conduct a semantic vector similarity search for 30 text blocks. Using a query composed of 50 random characters, we carry out both the hybrid search and semantic vector similarity search methods and record the time taken for each (this includes the embedding computation time). After repeating the experiment 100 times, we calculate the average time taken. The hybrid search method had an average duration of 0.0310 seconds, while the semantic vector similarity search took slightly less, with an average

time of 0.0305 seconds. This experiment aligns well with our theoretical analysis of the time complexity.

6.5 Conclusion

This chapter addressed the inherent challenges faced by semantic similarity search methods when linking LLMs to knowledge bases. Our method, AutoKG, presents a refined and efficient strategy for automated KG construction. In comparison to traditional KGs, the innovative architecture of AutoKG offers a lightweight and simplified version of KG, shifting the focus from specific entities to more abstract keywords and utilizing weighted undirected edges to represent the associations between keywords. Based on the generated KG, our approach harnesses these capabilities by presenting the LLMs with a more interconnected and comprehensive knowledge retrieval mechanism through the hybrid search strategy. By doing so, we ensure that the model's responses are not only richer in quality but also derive insights from a more diverse set of information nodes.

We tested AutoKG with a hybrid search in experimental evaluations. Because of dataset limitations, our tests were mostly qualitative. The outcome highlights the benefits of our method compared to typical RAG methods with semantic similarity search. In summary, AutoKG provides a valuable step to combine knowledge bases with LLMs. It is computationally lightweight and paves the way for more detailed interactions in LLM applications. Moreover, our hybrid search and the semantic vector similarity search have the same order of time complexity.

Further analysis of the *AutoKG* approach requires the identification or creation of an appropriate dataset to evaluate its integration with LLMs. Wang et al. [WLR23] developed their own dataset to evaluate a similar idea to ours. While the evaluation criteria should resemble that of RAG, a more structurally intricate and complex dataset is desired. Another avenue for improvement revolves around keyword extraction. Currently, the method

leverages prompt engineering; however, future work could explore fine-tuning larger models or even training specialized models to achieve enhanced results.

CHAPTER 7

Conclusion

This thesis has delved into various facets of graph-based learning and its applications, focusing on image analysis and extending to natural language processing. We have developed innovative methods and pipelines that enhance performance, minimize manual labeling costs, and broaden the applicability of these techniques across different domains.

In Chapter 3, we introduced two novel batch active learning methods, DAC for core-set selection and LocalMax for batch sampling, and their applications in SAR image classification and multi- and hyperspectral image segmentation. Our proposed methods exhibited exceptional performance in both domains, achieving high accuracy with a limited amount of labeled data while significantly improving efficiency compared to sequential active learning methods. Experimental results demonstrated that our methods outperformed state-of-the-art SAR classification methods on the OpenSARShip and FUSAR-Ship datasets, and achieved better overall accuracy with fewer labeled pixels in multi- and hyperspectral image segmentation tasks. The success of our DAC and LocalMax methods in these diverse applications highlighted the versatility and effectiveness of our batch active learning approach. Future research directions include extending these methods to other image analysis tasks and exploring the integration of more advanced feature embedding techniques.

Chapter 4 focused on graph-based active learning for surface water and sediment detection in multispectral images. We developed GAP and CGAP pipelines that significantly reduced manual labeling costs and improved the accuracy of water and sediment detection. The contrastive learning approach employed in CGAP offered robustness to different reso-

lutions, cloud coverage, and geometric transformations while also improving the efficiency of subsequent steps. Experiments showed that our methods surpassed the performance of CNN neural networks trained with 2.1 million pixels and outperformed SVM and RF models trained with larger datasets, using only around 3000 training vectors. We also provided a Python-based tool, GlobalRiverPIXELS, to detect surface water and sediment globally using our B-CGAP method. Future work could focus on enhancing the capability of models in classifying urban information and extending the models to more datasets.

In Chapter 5, we proposed two semi-supervised hyperspectral models, graph learning unmixing (GLU) and graph-regularized semi-supervised unmixing (GRSU), for nearly blind hyperspectral unmixing. Our methods demonstrated significant improvement with minimal supervision and the ability to work with either ground-truth abundance maps or one-hot pseudo labels. Extensive experiments using four standard hyperspectral datasets showed that our methods outperformed five state-of-the-art methods in blind hyperspectral unmixing. The proposed methods showed great potential for real-world problems, as they do not require a ground truth abundance map and can work with pseudo-labels instead. Future research could explore incorporating scaling factor and spectral variability terms into the models and replacing the current graph Laplacian learning solver with neural networks or graph neural network techniques.

Finally, Chapter 6 introduced AutoKG, an efficient method for automated knowledge graph (KG) generation for large language models (LLMs). This is not an application of image analysis but serves as an extended approach to graph learning. AutoKG provided a lightweight and simplified version of KG, focusing on abstract keywords and utilizing weighted undirected edges to represent associations between keywords. We also presented a hybrid search strategy that ensured the model's responses were richer in quality and derived insights from a more diverse set of information nodes. Experimental evaluations highlighted the benefits of our method compared to typical retrieval-augmented generation (RAG) methods with semantic similarity search. Future work could focus on identifying or creating an

appropriate dataset to evaluate the integration of AutoKG with LLMs and exploring fine-tuning or training specialized models for keyword extraction.

In conclusion, this thesis has made significant contributions to graph-based learning and its applications. The novel methods and pipelines developed throughout the chapters have demonstrated their potential to improve performance, reduce manual labeling costs, and expand the applicability of these techniques across various domains. As graph-based learning continues to evolve and find new applications, the work presented in this thesis provides a solid foundation for further advancements in the field. By leveraging the strengths of graph-based methods, active learning, and efficient batch selection, we have developed powerful frameworks for semi-supervised learning in image analysis and natural language processing tasks. These advancements contribute to reducing the reliance on large amounts of labeled data while still achieving high-quality results, paving the way for more efficient and effective learning in various real-world applications.

REFERENCES

[Abu21] Bilal Abu-Salih. "Domain-specific knowledge graphs: A survey." *Journal of Network and Computer Applications*, **185**:103076, 2021.

[AD] AFRL and DARPA. "Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset." `https://www.sdms.afrl.af.mil/index.php?collection=mstar`.

[AMN98] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions." *Journal of the ACM*, **45**(6):891–923, 1998.

[AMZ18] Julia M Arnold, Linda J Moore, and Edmund G Zelnio. "Blending synthetic and measured data using transfer learning for synthetic aperture radar (SAR) target classification." In *Algorithms for Synthetic Aperture Radar Imagery XXV*, volume 10647, pp. 48–57. SPIE, 2018.

[ASG23] Mayank Agarwal, Priyanka Sharma, and Ayan Goswami. "Analysing the applicability of ChatGPT, Bard, and Bing to generate reasoning-based multiple-choice questions in medical physiology." *Cureus*, **15**(6), 2023.

[ASP13] Konstantinos M Andreadis, Guy J-P Schumann, and Tamlin Pavelsky. "A simple global river bankfull width and depth database." *Water Resources Research*, **49**(10):7164–7168, 2013.

[AYK21] Akari Asai, Xinyan Yu, Jungo Kasai, and Hanna Hajishirzi. "One question answering model for many languages with cross-lingual dense passage retrieval." *Advances in Neural Information Processing Systems*, **34**:7547–7560, 2021.

[AZK20] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. "Deep batch active learning by diverse, uncertain gradient lower bounds." In *International Conference on Learning Representations*, 2020.

[BBT18] Zachary M. Boyd, Egil Bae, Xue-Cheng Tai, and Andrea L. Bertozzi. "Simplified energy landscape for modularity using total variation." *SIAM Journal on Applied Mathematics*, **78**(5):2439–2464, 2018.

[BCH23] Jason Brown, Bohan Chen, Harris Hardiman-Mostow, Adrien Weihs, Andrea L Bertozzi, and Jocelyn Chanussot. "Material identification in complex environments: Neural network approaches to hyperspectral image analysis." In *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–5. IEEE, 2023.

[BCL23] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. "A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity." In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 675–718, Nusa Dua, Bali, November 2023. Association for Computational Linguistics.

[BCM05] Antoni Buades, Bartomeu Coll, and J-M Morel. "A non-local algorithm for image denoising." In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 60–65. IEEE, 2005.

[BEV07] Xavier Bresson, Selim Esedoḡlu, Pierre Vandergheynst, Jean-Philippe Thiran, and Stanley Osher. "Fast global minimization of the active contour/snake model." *Journal of Mathematical Imaging and Vision*, **28**(2):151–167, 2007.

[BF12] Andrea L Bertozzi and Arjuna Flenner. "Diffuse interface models on graphs for classification of high dimensional data." *Multiscale Modeling & Simulation*, **10**(3):1090–1118, 2012.

[BHL21] Andrea L Bertozzi, Bamdad Hosseini, Hao Li, Kevin Miller, and Andrew M Stuart. "Posterior consistency of semi-supervised regression on graphs." *Inverse Problems*, **37**(10):105011, 2021.

[BJ20] Jignesh S Bhatt and Manjunath V Joshi. "Deep learning in hyperspectral unmixing: A review." In *IEEE International Geoscience and Remote Sensing Symposium*, pp. 2189–2192. IEEE, 2020.

[Bje07] David M Bjerklie. "Estimating the bankfull velocity and discharge for rivers using remotely sensed river morphology information." *Journal of Hydrology*, **341**(3-4):144–155, 2007.

[BKB16] K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst. "Song recommendation with non-negative matrix factorization and graph total variation." In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2439–2443, 2016.

[BKS21] Monika Bansal, Munish Kumar, Monika Sachdeva, and Ajay Mittal. "Transfer learning for image classification using VGG19: Caltech-101 image data set." *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2021.

[BLR04] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. "Semi-supervised learning using randomized mincuts." In *Proceedings of the Twenty-First International Conference on Machine Learning*, p. 13, 2004.

[BLS18] Andrea L Bertozzi, Xiyang Luo, Andrew M Stuart, and Konstantinos C Zygalakis. "Uncertainty quantification in the classification of high dimensional data." *SIAM/ASA Journal Uncertainty Quantification*, **6**(2):568–595, 2018.

[BM19] Andrea L Bertozzi and Ekaterina Merkurjev. "Graph-based optimization approaches for machine learning, uncertainty quantification and networks." In *Handbook of Numerical Analysis*, volume 20, pp. 503–531. Elsevier, 2019.

[BNA13] Paul D Bates, Jefferey C Neal, Douglas Alsdorf, and Guy J-P Schumann. "Observing global surface water flood dynamics." *The Earth'S Hydrological Cycle*, pp. 839–852, 2013.

[BOC23] Jason Brown, Riley O'Neill, Jeff Calder, and Andrea L Bertozzi. "Utilizing contrastive learning for graph-based active learning of SAR data." In *Algorithms for Synthetic Aperture Radar Imagery XXX*, volume 12520, pp. 181–195. SPIE, 2023.

[BPB20] Zachary M. Boyd, Mason A. Porter, and Andrea L. Bertozzi. "Stochastic block models are a discrete surface tension." *Journal of Nonlinear Science*, **30**(5):2429–2462, 2020.

[BPC13] José M Bioucas-Dias, Antonio Plaza, Gustavo Camps-Valls, Paul Scheunders, Nasser Nasrabadi, and Jocelyn Chanussot. "Hyperspectral remote sensing data analysis and future challenges." *IEEE Geoscience and Remote Sensing Magazine*, **1**(2):6–36, 2013.

[BPD12] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **5**(2):354–379, 2012.

[Cal18] Jeff Calder. "The game theoretic p-Laplacian and semi-supervised learning with few labels." *Nonlinearity*, **32**(1):301, 2018.

[Cal19] Jeff Calder. "Consistency of Lipschitz learning with infinite unlabeled data and finite labeled data." *SIAM Journal on Mathematics of Data Science*, **1**(4):780–812, 2019.

[Cal22] Jeff Calder. "GraphLearning Python Package.", January 2022.

[CB23] Bohan Chen and Andrea L Bertozzi. "AutoKG: Efficient automated knowledge graph generation for language models." In *2023 IEEE International Conference on Big Data (BigData)*, pp. 3117–3126. IEEE, 2023.

[CBC14] A Ciurte, X Bresson, O Cuisenaire, N Houhou, S Nedevschi, J-P Thiran, and Meritxell Bach Cuadra. "Semi-Supervised segmentation of ultrasound images based on patch representation and continuous min cut." *PLoS ONE*, **9**(7), 2014.

[CCT20] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. "Poisson learning: Graph based semi-supervised learning at very low label rates." In *International Conference on Machine Learning*, pp. 1306–1316. PMLR, 2020.

[CCT23] James Chapman, Bohan Chen, Zheng Tan, Jeff Calder, Kevin Miller, and Andrea L Bertozzi. "Novel batch active learning approach and its application on the synthetic aperture radar datasets." In *Algorithms for Synthetic Aperture Radar Imagery XXX*, volume 12520, pp. 96–111. SPIE, 2023.

[CH16] Gong Cheng and Junwei Han. "A survey on object detection in optical remote sensing images." *ISPRS Journal of Photogrammetry and Remote Sensing*, **117**:11–28, 2016.

[CK13] Yuxin Chen and Andreas Krause. "Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization." In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 160–168, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[CKN20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[CKS20] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. "Big self-supervised models are strong semi-supervised learners." *Advances in neural information processing systems*, **33**:22243–22255, 2020.

[CLB23] Bohan Chen, Yifei Lou, Andrea L. Bertozzi, and Jocelyn Chanussot. "Graph-based active learning for nearly blind hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **61**:1–16, 2023.

[CMB23a] Bohan Chen, Kevin Miller, Andrea L Bertozzi, and Jon Schwenk. "Batch active learning for multispectral and hyperspectral image segmentation using similarity graphs." *Communications on Applied Mathematics and Computation*, pp. 1–21, 2023.

[CMB23b] Bohan Chen, Kevin Miller, Andrea L. Bertozzi, and Jon Schwenk. "Graph-based active learning for surface water and sediment detection in multispectral images." In *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, pp. 5431–5434, 2023.

[CMB24] Bohan Chen, Kevin Miller, Andrea L. Bertozzi, and Jon Schwenk. "CGAP: A hybrid contrastive and graph-based active learning pipeline to detect water and sediment in multispectral images." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024. Submitted.

[CND23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. "Palm: Scaling language modeling with pathways." *Journal of Machine Learning Research*, **24**(240):1–113, 2023.

[CRS21] Sarah W Cooley, Jonathan C Ryan, and Laurence C Smith. "Human alteration of global surface water storage variability." *Nature*, **591**(7848):78–81, 2021.

[CS20] Jeff Calder and Dejan Slepčev. "Properly-weighted graph Laplacian for semi-supervised learning." *Applied Mathematics & Optimization*, **82**:1111–1159, 2020.

[CSK21] Tan Chen, Chunqiao Song, Linghong Ke, Jida Wang, Kai Liu, and Qianhan Wu. "Estimating seasonal water budgets in global lakes by using multi-source remote sensing measurements." *Journal of Hydrology*, **593**:125781, 2021.

[CTB13] Gustavo Camps-Valls, Devis Tuia, Lorenzo Bruzzone, and Jon Atli Benediktsson. "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods." *IEEE Signal Processing Magazine*, **31**(1):45–54, 2013.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." *Machine Learning*, **20**(3):273–297, 1995.

[CV01] Tony F Chan and Luminita A Vese. "Active contours without edges." *IEEE Transactions on Image Processing*, **10**(2):266–277, 2001.

[CYX20] Xiangyong Cao, Jing Yao, Zongben Xu, and Deyu Meng. "Hyperspectral image classification with convolutional neural network and active learning." *IEEE Transactions on Geoscience and Remote Sensing*, **58**(7):4604–4616, 2020.

[CZC17] HongYun Cai, Vincent Wenchen Zheng, and Kevin Chen-Chuan Chang. "Active Learning for Graph Embedding." *Computing Research Repository (CoRR)*, **abs/1705.05085**, 2017.

[CZP09] A. Cichocki, R. Zdunek, A. H. Phan, and S. I. Amari. *Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[Das11] Sanjoy Dasgupta. "Two faces of active learning." *Theoretical Computer Science*, **412**(19):1767–1781, 2011.

[DB21] Vijay S Deshpande, Jignesh S Bhatt, et al. "A practical approach for hyperspectral unmixing using deep learning." *IEEE Geoscience and Remote Sensing Letters*, **19**:1–5, 2021.

[DCL18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *ArXiv Preprint ArXiv*, 2018.

[DCS17] Ashraf Dewan, Robert Corner, Ashty Saleem, Md Masudur Rahman, Md Rafiqul Haider, Md Mostafizur Rahman, and Maminul H Sarker. "Assessing channel changes of the Ganges-Padma River system in Bangladesh using Landsat and hydrological data." *Geomorphology*, **276**:257–279, 2017.

[Dij59] E. W. Dijkstra. "A note on two problems in connexion with graphs." *Numerische Mathematik*, **1**(1):269–271, dec 1959.

[DMC19] Lucas Drumetz, Travis R Meyer, Jocelyn Chanussot, Andrea L Bertozzi, and Christian Jutten. "Hyperspectral image unmixing with endmember bundles and group sparsity inducing mixed norms." *IEEE Trans Image Process*, **28**(7):3435–3450, 2019.

[DVH16] Lucas Drumetz, Miguel-Angel Veganzones, Simon Henrot, Ronald Phlypo, Jocelyn Chanussot, and Christian Jutten. "Blind hyperspectral unmixing using an extended linear mixing model to address spectral variability." *IEEE Transactions on Image Processing*, **25**(8):3890–3905, 2016.

[EB92] Jonathan Eckstein and Dimitri P Bertsekas. "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators." *Mathematical Programming*, **55**:293–318, 1992.

[ECR16] Ahmed El Alaoui, Xiang Cheng, Aaditya Ramdas, Martin J Wainwright, and Michael I Jordan. "Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning." In *Conference on Learning Theory*, pp. 879–906. PMLR, 2016.

[EH90] Alan Egger and Robert Huotari. "Rate of convergence of the discrete Pólya algorithm." *Journal of Approximation Theory*, **60**(1):24–30, 1990.

[Eva22] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[FBC04] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. "Spectral grouping using the Nystrom method." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(2):214–225, 2004.

[FCL22] Mauricio Flores, Jeff Calder, and Gilad Lerman. "Analysis and algorithms for ℓ_p-based semi-supervised learning on graphs." *Applied and Computational Harmonic Analysis*, **60**:77–122, 2022.

[FLW22] Xin-Ru Feng, Heng-Chao Li, Rui Wang, Qian Du, Xiuping Jia, and Antonio Plaza. "Hyperspectral unmixing based on nonnegative matrix factorization: A comprehensive review." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **15**:4414–4436, 2022.

[Gao96] Bo-Cai Gao. "NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space." *Remote Sensing of Environment*, **58**(3):257–266, 1996.

[GB10] Andrew Guillory and Jeff Bilmes. "Interactive submodular set cover." In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, p. 415–422, Madison, WI, USA, 2010. Omnipress.

[GIG17] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. "Deep Bayesian active learning with image data." In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 1183–1192. JMLR.org, 2017.

[GMB14] Cristina Garcia-Cardona, Ekaterina Merkurjev, Andrea L. Bertozzi, Arjuna Flenner, and Allon G. Percus. "Multiclass data segmentation using diffuse interface methods on graphs." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**(8):1600–1613, 2014.

[GO09] Guy Gilboa and Stanley Osher. "Nonlocal operators with applications to image processing." *Multiscale Modeling & Simulation*, **7**(3):1005–1028, 2009.

[Gon09] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009.

[GS96] Ralph Grishman and Beth M Sundheim. "Message understanding conference-6: A brief history." In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[Has] Mahdi Hasanlou. "Remote Sensing Datasets." `https://rslab.ut.ac.ir/data`. Accessed: 2023-3-10.

[HAS20] Xiyue Hou, Wei Ao, Qian Song, Jian Lai, Haipeng Wang, and Feng Xu. "FUSAR-Ship: Building a high-resolution SAR-AIS matchup dataset of Gaofen-3 for ship detection and recognition." *Science China Information Sciences*, **63**(4):1–19, 2020.

[Hay98] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[Hei01] Daniel C Heinz et al. "Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery." *IEEE Transactions on Geoscience and Remote Sensing*, **39**(3):529–545, 2001.

[HGY21] Danfeng Hong, Lianru Gao, Jing Yao, Naoto Yokoya, Jocelyn Chanussot, Uta Heiden, and Bing Zhang. "Endmember-guided unmixing network (EGU-Net): A general deep learning framework for self-supervised hyperspectral unmixing." *IEEE Transactions on Neural Networks and Learning Systems*, **33**(11):6518–6531, 2021.

[HHG22] Zhu Han, Danfeng Hong, Lianru Gao, Jing Yao, Bing Zhang, and Jocelyn Chanussot. "Multimodal hyperspectral unmixing: Insights from attention networks." *IEEE Transactions on Geoscience and Remote Sensing*, **60**:1–13, 2022.

[HLL17] Lanqing Huang, Bin Liu, Boying Li, Weiwei Guo, Wenhao Yu, Zenghui Zhang, and Wenxian Yu. "OpenSARShip: A dataset dedicated to Sentinel-1 ship interpretation." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **11**(1):195–208, 2017.

[HLP13] Huiyi Hu, Thomas Laurent, Mason A. Porter, and Andrea L. Bertozzi. "A Method based on total variation for network modularity optimization using the MBO scheme." *SIAM Journal on Applied Mathematics*, **73**(6):2224–2246, 2013.

[Ho95] Tin Kam Ho. "Random decision forests." In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pp. 278–282. IEEE, 1995.

[HPG14] R. Heylen, M. Parente, and P. Gader. "A review of nonlinear hyperspectral unmixing methods." *Remote Sensing*, **7**(6):1844–1868, 2014.

[HSB15] Huiyi Hu, Justin Sunu, and Andrea L Bertozzi. "Multi-class graph Mumford-Shah model for plume detection using the MBO scheme." In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 209–222. Springer, 2015.

[HSC23] Matthew Ho, Aditya Sharma, Justin Chang, Michael Saxon, Sharon Levy, Yujie Lu, and William Yang Wang. "WikiWhy: Answering and explaining cause-and-effect questions." In *The Eleventh International Conference on Learning Representations*, 2023.

[HWW20] Luyang Huang, Lingfei Wu, and Lu Wang. "Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward." In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5094–5107, Online, July 2020. Association for Computational Linguistics.

[HYC19a] Danfeng Hong, Naoto Yokoya, Jocelyn Chanussot, Jian Xu, and Xiao Xiang Zhu. "Learning to propagate labels on graphs: An iterative multitask regression framework for semi-supervised hyperspectral dimensionality reduction." *ISPRS Journal of Photogrammetry and Remote Sensing*, **158**:35–49, 2019.

[HYC19b] Danfeng Hong, Naoto Yokoya, Jocelyn Chanussot, and Xiao Xiang Zhu. "An augmented linear mixing model to address spectral variability for hyperspectral unmixing." *IEEE Transactions on Image Processing*, **28**(4):1923–1938, 2019.

[HYG19] Danfeng Hong, Naoto Yokoya, Nan Ge, Jocelyn Chanussot, and Xiao Xiang Zhu. "Learnable manifold alignment (LeMA): A semi-supervised cross-modality learning framework for land cover and land use classification." *ISPRS Journal of Photogrammetry and Remote Sensing*, **147**:193–205, 2019.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[HZR22] Hangfeng He, Hongming Zhang, and Dan Roth. "Rethinking with retrieval: Faithful large language model inference." *ArXiv Preprint ArXiv*, 2022.

[HZZ16] W. He, H. Zhang, and L. Zhang. "Sparsity-regularized robust non-negative matrix factorization for hyperspectral unmixing." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(9):4267–4279, 2016.

[HZZ17] W. He, H. Zhang, and L. Zhang. "Total variation regularized reweighted sparse nonnegative matrix factorization for hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **55**(7):3909–3921, 2017.

[IBP11] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza. "Sparse unmixing of hyperspectral data." *IEEE Transactions on Geoscience and Remote Sensing*, **49**(6):2014–2039, 2011.

[IBP12] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza. "Total variation spatial regularization for sparse hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **50**(11):4484–4502, 2012.

[IBP19] Leo F Isikdogan, Alan Bovik, and Paola Passalacqua. "Seeing through the clouds with deepwatermap." *IEEE Geoscience and Remote Sensing Letters*, **17**(10):1662–1666, 2019.

[ICB21] Geoffrey Iyer, Jocelyn Chanussot, and Andrea L. Bertozzi. "A Graph-Based approach for data fusion and segmentation of multimodal images." *IEEE Transactions on Geoscience and Remote Sensing*, **59**(5):4419–4429, May 2021.

[IID21] Nathan Inkawhich, Matthew J Inkawhich, Eric K Davis, Uttam K Majumder, Erin Tripp, Chris Capraro, and Yiran Chen. "Bridging a gap in SAR-ATR: Training on fully synthetic and testing on measured data." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **14**:2942–2955, 2021.

[JH12] Ming Ji and Jiawei Han. "A variance minimization criterion to active learning on graphs." In *Artificial Intelligence and Statistics*, pp. 556–564. PMLR, 2012.

[JLF23] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. "Survey of hallucination in natural language generation." *ACM Computing Surveys*, **55**(12):1–38, 2023.

[JMG22] Hamid Jafarzadeh, Masoud Mahdianpari, and Eric Gill. "Wet-GC: A novel multi-model graph convolutional approach for wetland classification using Sentinel-1 and 2 imagery with limited training samples." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2022.

[Jon19] John W Jones. "Improved automated detection of subpixel-scale inundation—Revised dynamic surface water extent (DSWE) partial surface water tests." *Remote Sensing*, **11**(4):374, 2019.

[JPC21] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. "A survey on knowledge graphs: Representation, acquisition, and applications." *IEEE Transactions on Neural Networks and Learning Systems*, **33**(2):494–514, 2021.

[JZL19] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. "Semi-supervised learning with graph learning-convolutional networks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11313–11320, 2019.

[KLS17] Nataliia Kussul, Mykola Lavreniuk, Sergii Skakun, and Andrii Shelestov. "Deep learning classification of land cover and crop types using remote sensing data." *IEEE Geoscience and Remote Sensing Letters*, **14**(5):778–782, 2017.

[KPR21] Vidya Kandekar, Chaitanya Pande, Jayaraman Rajesh, AA Atre, SD Gorantiwar, SA Kadam, Bhau Gavit, et al. "Surface water dynamics analysis based on sentinel imagery and Google Earth Engine Platform: a case study of Jayakwadi dam." *Sustainable Water Resources Management*, **7**(3):1–11, 2021.

[KRS15] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A Spielman. "Algorithms for Lipschitz learning on graphs." In *Conference on Learning Theory*, pp. 1190–1223. PMLR, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in Neural Information Processing Systems*, **25**, 2012.

[KTW20] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. "Supervised contrastive learning." *Advances in neural information processing systems*, **33**:18661–18673, 2020.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *ArXiv Preprint ArXiv*, 2013.

[KW17] Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In *International Conference on Learning Representations*, 2017.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. "Snakes: Active contour models." *International Journal of Computer Vision*, **1**(4):321–331, 1988.

[LBD89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition." *Neural Computation*, **1**(4):541–551, 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature*, **521**(7553):436–444, 2015.

[LBP10] Jun Li, José M. Bioucas-Dias, and Antonio Plaza. "Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning." *IEEE Transactions on Geoscience and Remote Sensing*, **48**(11):4085–4098, 2010.

[LET21] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. "Sparse, dense, and attentional representations for text retrieval." *Transactions of the Association for Computational Linguistics*, **9**:329–345, 2021.

[LLH23] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. "Normalizing flow-based neural process for few-shot knowledge graph completion." In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, p. 900–910, New York, NY, USA, 2023. Association for Computing Machinery.

[Llo82] Stuart Lloyd. "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, **28**(2):129–137, 1982.

[LOG19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized Bert pretraining approach." *ArXiv Preprint ArXiv*, 2019.

[LPA20] Peirong Lin, Ming Pan, George H Allen, Renato Prata de Frasson, Zhenzhong Zeng, Dai Yamazaki, and Eric F Wood. "Global estimates of reach-level bankfull river width leveraging big data geospatial analysis." *Geophysical Research Letters*, **47**(7):e2019GL086405, 2020.

[LPP20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems*, **33**:9459–9474, 2020.

[LS99] D. D. Lee and H. S. Seung. "Learning the parts of objects by non-negative matrix factorization." *Nature*, **401**(6755):788, 1999.

[LSF19] Shutao Li, Weiwei Song, Leyuan Fang, Yushi Chen, Pedram Ghamisi, and Jon Atli Benediktsson. "Deep learning for hyperspectral image classification: An overview." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(9):6690–6709, 2019.

[LSS19] Benjamin Lewis, Theresa Scarnati, Elizabeth Sudkamp, John Nehrbass, Stephen Rosencrantz, and Edmund Zelnio. "A SAR dataset for ATR development: the Synthetic and Measured Paired Labeled Experiment (SAMPLE)." In *Algorithms for Synthetic Aperture Radar Imagery XXVI*, volume 10987, pp. 39–54. SPIE, 2019.

[LWW11] Yunmei Li, Qiao Wang, Chuanqing Wu, Shaohua Zhao, Xing Xu, Yanfei Wang, and Changchun Huang. "Estimation of chlorophyll a concentration using NIR/red bands of MERIS and classification procedure in inland turbid water." *IEEE Transactions on Geoscience and Remote Sensing*, **50**(3):988–997, 2011.

[LWY12] Xiaoqiang Lu, Hao Wu, Yuan Yuan, Pingkun Yan, and Xuelong Li. "Manifold regularized sparse NMF for hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **51**(5):2815–2826, 2012.

[LZG19] M. Li, F. Zhu, A. J. X. Guo, and J. Chen. "A graph regularized multilinear mixing model for nonlinear hyperspectral unmixing." *Remote Sensing*, **11**(19):2188, 2019.

[Mac67] James MacQueen et al. "Some methods for classification and analysis of multivariate observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297. Oakland, CA, USA, 1967.

[MB24] Kevin S Miller and Andrea L Bertozzi. "Model change active learning in graph-based semi-supervised learning." *Communications on Applied Mathematics and Computation*, pp. 1–29, 2024.

[MBS09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. "Distant supervision for relation extraction without labeled data." In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, 2009.

[MC23] Kevin Miller and Jeff Calder. "Poisson reweighted Laplacian uncertainty sampling for graph-based active learning." *SIAM Journal on Mathematics of Data Science*, **5**(4):1160–1190, 2023.

[McF96] Stuart K McFeeters. "The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features." *International Journal of Remote Sensing*, **17**(7):1425–1432, 1996.

[MCH18] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. "Never-ending learning." *Communications of the ACM*, **61**(5):103–115, 2018.

[MDL23] Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. "Augmented Language Models: A Survey." *Transactions on Machine Learning Research*, 2023. Survey Certification.

[Mer94] Russell Merris. "Laplacian matrices of graphs: A survey." *Linear Algebra and its Applications*, **197**:143–176, 1994.

[MGB14] Ekaterina Merkurjev, Cristina Garcia-Cardona, Andrea L Bertozzi, Arjuna Flenner, and Allon G Percus. "Diffuse interface methods for multiclass segmentation of high-dimensional data." *Applied Mathematics Letters*, **33**:29–34, 2014.

[MH08] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of Machine Learning Research*, **9**(11), 2008.

[MHS15] Yifei Ma, Tzu-Kuo Huang, and Jeff G Schneider. "Active Search and Bandits on Graphs using Sigma-Optimality." In *Proceedings of 31st Conference on Uncertainty in Artificial Intelligence (UAI '15)*, volume 542, p. 551, 2015.

[MHS18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. "UMAP: Uniform Manifold Approximation and Projection." *Journal of Open Source Software*, **3**(29):861, 2018.

[MKB13] Ekaterina Merkurjev, Tijana Kostic, and Andrea L Bertozzi. "An MBO scheme on graphs for classification and image processing." *SIAM Journal on Imaging Sciences*, **6**(4):1903–1930, 2013.

[MKS13] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. "Distributed submodular maximization: Identifying representative elements in massive data." *Advances in Neural Information Processing Systems*, **26**, 2013.

[MLB20] Kevin Miller, Hao Li, and Andrea L. Bertozzi. "Efficient graph-based active learning with probit likelihood via Gaussian approximations." *Computing Research Repository (CoRR)*, **abs/2007.11126**, 2020.

[MLZ19] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. "Deep learning in remote sensing applications: A meta-analysis and review." *ISPRS Journal of Photogrammetry and Remote Sensing*, **152**:166–177, 2019.

[MM19] James M. Murphy and Mauro Maggioni. "Unsupervised clustering and active learning of hyperspectral images with nonlinear diffusion." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(3):1829–1845, 2019.

[MMK17] Zhaoyi Meng, Ekaterina Merkurjev, Alice Koniges, and Andrea L Bertozzi. "Hyperspectral image classification using graph clustering methods." *Image Processing On Line*, **7**:218–245, 2017.

[MMS22] Kevin Miller, Jack Mauro, Jason Setiadi, Xoaquin Baca, Zhan Shi, Jeff Calder, and Andrea L Bertozzi. "Graph-based active learning for semi-supervised classification of SAR data." In *Algorithms for Synthetic Aperture Radar Imagery XXIX*, volume 12095, pp. 126–139. SPIE, 2022.

[MS89] David Bryant Mumford and Jayant Shah. "Optimal approximations by piecewise smooth functions and associated variational problems." *Communications on Pure and Applied Mathematics*, 1989.

[MSB14] Ekaterina Merkurjev, Justin Sunu, and Andrea L Bertozzi. "Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video." In *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 689–693. IEEE, 2014.

[MSP20] Isaiah Onando Mulang', Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. "Evaluating the impact of knowledge graph context on entity disambiguation models." In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, p. 2157–2160, New York, NY, USA, 2020. Association for Computing Machinery.

[Mul13] David J Mulla. "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps." *Biosystems Engineering*, **114**(4):358–371, 2013.

[MWV12] Freek D Van der Meer, Harald MA Van der Werff, Frank JA Van Ruitenbeek, Chris A Hecker, Wim H Bakker, Marleen F Noomen, Mark Van Der Meijde, E John M Carranza, J Boudewijn De Smeth, and Tsehaie Woldai. "Multi- and hyperspectral geologic remote sensing: A review." *International Journal of Applied Earth Observation and Geoinformation*, **14**(1):112–128, 2012.

[ND05] José MP Nascimento and José MB Dias. "Vertex component analysis: A fast algorithm to unmix hyperspectral data." *IEEE Transactions on Geoscience and Remote Sensing*, **43**(4):898–910, 2005.

[New03] Mark EJ Newman. "The structure and function of complex networks." *SIAM Review*, **45**(2):167–256, 2003.

[NJW01] Andrew Ng, Michael Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." *Advances in Neural Information Processing Systems*, **14**, 2001.

[NS07] David Nadeau and Satoshi Sekine. "A survey of named entity recognition and classification." *Lingvisticae Investigationes*, **30**(1):3–26, 2007.

[NSZ09] Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. "Semi-supervised learning with the graph Laplacian: The limit of infinite unlabelled data." *Advances in Neural Information Processing Systems*, **22**:1330–1338, 2009.

[NWF78] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I." *Mathematical Programming*, **14**:265–294, 1978.

[OKA18] Savas Ozkan, Berk Kaya, and Gozde Bozdagi Akar. "Endnet: Sparse autoencoder network for endmember extraction and hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(1):482–496, 2018.

[OWO14] Braxton Osting, Chris D White, and Édouard Oudet. "Minimal Dirichlet energy partitions for graphs." *Journal of Scientific Computing*, **36**(4):A1635–A1651, 2014.

[PCG16] Jean-François Pekel, Andrew Cottam, Noel Gorelick, and Alan S Belward. "High-resolution mapping of global surface water and its long-term changes." *Nature*, **540**(7633):418–422, 2016.

[PGH16] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. "Variational autoencoder for deep learning of images, labels and captions." *Advances in Neural Information Processing Systems*, **29**, 2016.

[PLW23] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. "Unifying large language models and knowledge graphs: A roadmap." *ArXiv Preprint ArXiv*, 2023.

[PPP06] V. P. Pauca, J. Piper, and R. J. Plemmons. "Nonnegative matrix factorization for spectral data analysis." *Linear Algebra and its Applications*, **416**(1):29–47, 2006.

[PRL19] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. "Language models as knowledge bases?" *ArXiv Preprint ArXiv*, 2019.

[PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, **12**:2825–2830, 2011.

[PWS16] Nathalie Pettorelli, Martin Wegmann, Andrew Skidmore, Sander Mücher, Terence P Dawson, Miguel Fernandez, Richard Lucas, Michael E Schaepman, Tiejun Wang, Brian O'Connor, et al. "Framing the concept of satellite remote sensing essential biodiversity variables: challenges and future directions." *Remote Sensing in Ecology and Conservation*, **2**(3):122–131, 2016.

[PY09] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on Knowledge and Data Engineering*, **22**(10):1345–1359, 2009.

[QJZ11] Y. Qian, S. Jia, J. Zhou, and A. Robles-Kelly. "Hyperspectral unmixing via $L_{1/2}$ sparsity-constrained nonnegative matrix factorization." *IEEE Transactions on Geoscience and Remote Sensing*, **49**(11):4282–4297, 2011.

[QLC19] Jing Qin, Harlin Lee, Jocelyn T Chi, Yifei Lou, Jocelyn Chanussot, and Andrea L Bertozzi. "Fast blind hyperspectral unmixing based on graph Laplacian." In *Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–5. IEEE, 2019.

[QLC21] Jing Qin, Harlin Lee, Jocelyn T. Chi, Lucas Drumetz, Jocelyn Chanussot, Yifei Lou, and Andrea L. Bertozzi. "Blind hyperspectral unmixing based on graph total variation Regularization." *IEEE Transactions on Geoscience and Remote Sensing*, **59**(4):3338–3351, 2021.

[QSW19] Yiling Qiao, Chang Shi, Chenjian Wang, Hao Li, Matt Haberland, Xiyang Luo, Andrew M Stuart, and Andrea L Bertozzi. "Uncertainty quantification for semi-supervised multi-class classification in image processing and ego-motion analysis of body-worn videos." *Electronic Imaging*, **2019**(11):264–1, 2019.

[Ram03] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries." In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pp. 29–48. Citeseer, 2003.

[RF17] Mattia Rossi and Pascal Frossard. "Graph-based light field super-resolution." In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6. IEEE, 2017.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

[RGC08] Suju Rajan, Joydeep Ghosh, and Melba M. Crawford. "An active learning approach to hyperspectral data classification." *IEEE Transactions on Geoscience and Remote Sensing*, **46**(4):1231–1242, 2008.

[RKS22] Behnood Rasti, Bikram Koirala, Paul Scheunders, and Jocelyn Chanussot. "Misicnet: Minimum simplex convolutional network for deep hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **60**:1–15, 2022.

[RNC14] Max G Rozo, Afonso CR Nogueira, and Carlomagno Soto Castro. "Remote sensing-based analysis of the planform changes in the Upper Amazon River over the period 1986–2006." *Journal of South American Earth Sciences*, **51**:28–44, 2014.

[ROF92] L. I. Rudin, S. J. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms." *Physica D: Nonlinear Phenomena*, **60**(1-4):259–268, 1992.

[RRT21] Antonio M Rinaldi, Cristiano Russo, and Cristian Tommasino. "A semantic approach for document classification using deep neural networks and multimedia knowledge graph." *Expert Systems with Applications*, **169**:114320, 2021.

[RSP16] Joel C Rowland, Eitan Shelef, Paul A Pope, Jordan Muss, Chandana Gangodagamage, Steven P Brumby, and Cathy J Wilson. "A morphology independent methodology for quantifying planview river change and characteristics from remotely sensed imagery." *Remote Sensing of Environment*, **184**:212–228, 2016.

[RSR20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." *Journal of Machine Learning Research*, **21**(1):5485–5551, 2020.

[SB88] Gerard Salton and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." *Information Processing & Management*, **24**(5):513–523, 1988.

[SCM22] Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. "Fine-tuned language models are continual learners." In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[SDB03] P. Sajda, S. Du, T. Brown, L. Parra, and R. Stoyanova. "Recovery of constituent spectra in 3D chemical shift imaging using nonnegative matrix factorization." In *4th International Symposium on Independent Component Analysis and Blind Signal Separation*, pp. 71–76, 2003.

[Set09] Burr Settles. "Active learning literature survey." 2009.

[Sha48] Claude Elwood Shannon. "A mathematical theory of communication." *The Bell System Technical Journal*, **27**(3):379–423, 1948.

[SJH15] Michael E Schaepman, Michael Jehle, Andreas Hueni, Petra D'Odorico, Alexander Damm, Jürg Weyermann, Fabian D Schneider, Valérie Laurent, Christoph Popp, Felix C Seidel, et al. "Advanced radiometry measurements and Earth science applications with the Airborne Prism Experiment (APEX)." *Remote Sensing of Environment*, **158**:207–219, 2015.

[SKF17] Jon Schwenk, Ankush Khandelwal, Mulu Fratkin, Vipin Kumar, and Efi Foufoula-Georgiou. "High spatiotemporal resolution of river planform dynamics from Landsat: The RivMAP toolbox and results from the Ucayali River." *Earth and Space Science*, **4**(2):46–75, 2017.

[SM14] A. Sandryhaila and J. M. Moura. "Discrete signal processing on graphs: Frequency analysis." *IEEE Transactions on Signal Processing*, **62**(12):3042–3054, 2014.

[SNF13] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." *IEEE Signal Processing Magazine*, **30**(3):83–98, 2013.

[SPR20] Jon Schwenk, Anastasia Piliouras, and Joel C Rowland. "Determining flow directions in river channel networks using planform morphology and topology." *Earth Surface Dynamics*, **8**(1):87–102, 2020.

[SR22] Jon Schwenk and Joel Rowland. "RiverPIXELS: paired Landsat images and expert-labeled sediment and water pixels for a selection of rivers v1.0." 1 2022.

[SS18] Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach." In *International Conference on Learning Representations*, 2018.

[Str01] Steven H Strogatz. "Exploring complex networks." *Nature*, **410**(6825):268–276, 2001.

[TBK22] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions." *ArXiv Preprint ArXiv*, 2022.

[TM13] Hossein Talebi and Peyman Milanfar. "Global image denoising." *IEEE Transactions on Image Processing*, **23**(2):755–768, 2013.

[TMS23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models." *ArXiv Preprint ArXiv*, 2023.

[TMZ22] Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. "Memorization without overfitting: Analyzing the training dynamics of large language models." *Advances in Neural Information Processing Systems*, **35**:38274–38290, 2022.

[TNX21] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. "GRAND++: Graph neural diffusion with a source term." In *International Conference on Learning Representations*, 2021.

[TSW23] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. "Graph neural prompting with large language models." *ArXiv Preprint ArXiv*, 2023.

[VBC20] SS Vijayashekhar, Jignesh S Bhatt, and Bhargab Chattopadhyay. "Virtual dimensionality of hyperspectral data: Use of multiple hypothesis testing for controlling type-I error." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **13**:2974–2985, 2020.

[VFM20] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. "Deep graph-convolutional image denoising." *IEEE Transactions on Image Processing*, **29**:8226–8237, 2020.

[VMG10] Charles J Vörösmarty, Peter B McIntyre, Mark O Gessner, David Dudgeon, Alexander Prusevich, Pamela Green, Stanley Glidden, Stuart E Bunn, Caroline A Sullivan, C Reidy Liermann, et al. "Global threats to human water security and river biodiversity." *Nature*, **467**(7315):555–561, 2010.

[Von07] Ulrike Von Luxburg. "A tutorial on spectral clustering." *Statistics and Computing*, **17**(4):395–416, 2007.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *Advances in Neural Information Processing Systems*, **30**, 2017.

[WC13] Weiran Wang and Miguel A Carreira-Perpinán. "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application." *ArXiv Preprint ArXiv*, 2013.

[Wes01] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[WIB15] Kai Wei, Rishabh Iyer, and Jeff Bilmes. "Submodularity in data subset selection and active learning." In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1954–1963, Lille, France, 07–09 Jul 2015. PMLR.

[WKR20] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. "Neural text generation with unlikelihood training." In *International Conference on Learning Representations*, 2020.

[WLR23] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. "Knowledge graph prompting for multi-document question answering." *ArXiv Preprint ArXiv*, 2023.

[WMW17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. "Knowledge graph embedding: A survey of approaches and applications." *IEEE Transactions on Knowledge and Data Engineering*, **29**(12):2724–2743, 2017.

[Woo50] Max A Woodbury. *Inverting modified matrices*. Department of Statistics, Princeton University, 1950.

[WPC20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." *IEEE Transactions on Neural Networks and Learning Systems*, **32**(1):4–24, 2020.

[WPG21] Guojia Wan, Shirui Pan, Chen Gong, Chuan Zhou, and Gholamreza Haffari. "Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning." In *International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence, 2021.

[WSZ19] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. "Simplifying graph convolutional networks." In *International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

[WY13] Zheng Wang and Jieping Ye. "Querying discriminative and representative samples for batch mode active learning." In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, p. 158–166, New York, NY, USA, 2013. Association for Computing Machinery.

[XQZ18] F. Xiong, Y. Qian, J. Zhou, and Y. Y. Tang. "Hyperspectral unmixing via total variation regularized nonnegative tensor factorization." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(4):2341–2357, 2018.

[XSY08] Yichun Xie, Zongyao Sha, and Mei Yu. "Remote sensing imagery in vegetation mapping: a review." *Journal of Plant Ecology*, **1**(1):9–23, 2008.

[XYC19] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. "Improving Question Answering over Incomplete KBs with Knowledge-Aware Reader." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4258–4264, Florence, Italy, July 2019. Association for Computational Linguistics.

[YBR22] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. "Deep bidirectional language-knowledge graph pretraining." *Advances in Neural Information Processing Systems*, **35**:37309–37323, 2022.

[YRH21] Yanyang Yan, Wenqi Ren, Xiaobin Hu, Kun Li, Haifeng Shen, and Xiaochun Cao. "SRGAT: Single image super-resolution with graph attention network." *IEEE Transactions on Image Processing*, **30**:4905–4918, 2021.

[YZW20] Yuan Yuan, Zihan Zhang, and Qi Wang. "Improved collaborative non-negative matrix factorization and total variation for hyperspectral unmixing." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **13**:998–1010, 2020.

[ZCH20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications." *AI Open*, **1**:57–81, 2020.

[ZCZ21] Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. "Neural, symbolic and neural-symbolic reasoning on knowledge graphs." *AI Open*, **2**:14–35, 2021.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. "Semi-supervised learning using Gaussian fields and harmonic functions." In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 912–919, 2003.

[Zhu05] Xiaojin Jerry Zhu. "Semi-supervised learning literature survey." 2005.

[Zhu17] Zhe Zhu. "Change detection using Landsat time series: A review of frequencies, preprocessing, algorithms, and applications." *ISPRS Journal of Photogrammetry and Remote Sensing*, **130**:370–384, 2017.

[ZLF19] Lina Zhuang, Chia-Hsiang Lin, Mario AT Figueiredo, and Jose M Bioucas-Dias. "Regularization parameter selection in minimum volume hyperspectral unmixing." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(12):9858–9877, 2019.

[ZLG03] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions." In *ICML 2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, volume 3, 2003.

[ZS05] Dengyong Zhou and Bernhard Schölkopf. "Regularization on discrete spaces." In *Joint Pattern Recognition Symposium*, pp. 361–368. Springer, 2005.

[ZTM17] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. "Deep learning in remote sensing: A comprehensive review and list of resources." *IEEE Geoscience and Remote Sensing Magazine*, **5**(4):8–36, 2017.

[ZWL23] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. "A comprehensive survey on automatic knowledge graph construction." *ACM Computing Surveys*, **56**(4), nov 2023.

[ZWX14] Feiyun Zhu, Ying Wang, Shiming Xiang, Bin Fan, and Chunhong Pan. "Structured sparse method for hyperspectral unmixing." *ISPRS Journal of Photogrammetry and Remote Sensing*, **88**:101–118, 2014.

[ZWZ22] Bing Zhang, Yuanfeng Wu, Boya Zhao, Jocelyn Chanussot, Danfeng Hong, Jing Yao, and Lianru Gao. "Progress and challenges in intelligent remote sensing satellite systems." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **15**:1814–1822, 2022.

[ZYL22] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. "Learning to prompt for vision-language models." *International Journal of Computer Vision*, **130**(9):2337–2348, 2022.

[ZZD16] Liangpei Zhang, Lefei Zhang, and Bo Du. "Deep learning for remote sensing data: A technical tutorial on the state of the art." *IEEE Geoscience and Remote Sensing Magazine*, **4**(2):22–40, 2016.

[ZZK21] Tianwen Zhang, Xiaoling Zhang, Xiao Ke, Chang Liu, Xiaowo Xu, Xu Zhan, Chen Wang, Israr Ahmad, Yue Zhou, Dece Pan, et al. "HOG-ShipCLSNet: A novel deep learning network with hog feature fusion for SAR ship classification." *IEEE Transactions on Geoscience and Remote Sensing*, **60**:1–22, 2021.