UCLA
UCLA Electronic Theses and Dissertations

Title

Label-Efficient Machine Learning: Structure-Driven Semi-Supervised Learning on Graphs and Self-Supervised Disturbance Mapping

Permalink

https://escholarship.org/uc/item/4gr0k9sb

ISBN

9798247921141

Author

Hardiman-Mostow, Harris Howard

Publication Date

2026-05-26

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Label-Efficient Machine Learning: Structure-Driven Semi-Supervised Learning on Graphs and Self-Supervised Disturbance Mapping

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

by

Harris Howard Hardiman-Mostow

2026

© Copyright by
Harris Howard Hardiman-Mostow
2026

ABSTRACT OF THE DISSERTATION

Label-Efficient Machine Learning: Structure-Driven Semi-Supervised Learning on Graphs and Self-Supervised Disturbance Mapping

by

Harris Howard Hardiman-Mostow
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2026
Professor Andrea L. Bertozzi, Chair

This dissertation develops novel *semi-supervised* and *self-supervised* learning algorithms, motivated by settings where labeled training data is scarce or unavailable, respectively. We study two core problems in graph-based semi-supervised learning: how to construct a meaningful graph, and how to exploit its structure. To address the former, we derive back-propagation equations to precisely integrate graph construction and graph-based learning into a neural network, so end-to-end training directly learns data embeddings conducive to high-quality graphs. Our novel Graph Learning Layer (GLL) demonstrates improved generalization and adversarial robustness compared to the standard softmax classification head across vision datasets, architectures, and label rates. To tackle the latter, we leverage graph curvature to design coreset selection and active learning routines that dynamically target different regions of the graph according to its topology. We also present an algorithm that modifies the graph's multiscale connectivity based on active learning acquisitions, enhancing local structure. On a variety of benchmark datasets, our methods consistently outperform existing work in coreset quality, sequential and batch active learning, and computational

efficiency, reaching higher accuracy faster and with fewer labels. Extending the theme of label-efficiency to disaster response with remote sensing imagery, we design a self-supervised vision transformer trained on near-global, time-series synthetic aperture radar (SAR) data to detect and delineate land disturbances. On three distinct real-world natural disasters, our approach yields accurate delineations and surpasses prior SAR disturbance methods, representing an important step towards operational and label-free disturbance monitoring using modern artificial intelligence tools.

The dissertation of Harris Howard Hardiman-Mostow is approved.

Jeff Calder

Hayden Schaeffer

Deanna M. Needell

Andrea L. Bertozzi, Committee Chair

University of California, Los Angeles

2026

To Mom and Dad

"Flap your wings!"

TABLE OF CONTENTS

1 Introduction **1**

1.1 Overview of the Dissertation 1

1.2 Graph-Based Semi-Supervised Learning 5

1.2.1 Problem Formulation and Notation 5

1.2.2 Graph Construction 6

1.2.3 Laplace Learning 6

1.2.4 Additional Background 9

2 GLL: A Differentiable Graph Learning Layer for Neural Networks . . . **12**

2.1 Introduction 13

2.1.1 Related Work 14

2.1.2 Our Contributions 14

2.2 Background 17

2.2.1 Similarity Graph Construction 17

2.2.2 Calculus on graphs 18

2.2.3 Classical Graph-based Learning 20

2.2.4 Diagonal Perturbation 22

2.2.5 p-Laplace learning 23

2.2.6 Poisson learning 23

2.3 Automatic differentiation through graph learning 24

2.3.1 Backpropagation through graph Laplace equations 25

2.3.2 Backpropagation through similarity matrices 35

2.4 Implementation & Numerical Experiments 40
2.4.1 Implementation 41
2.4.2 Understanding Embeddings and the Effect of Tikhonov Regularization via Two Moons 46
2.4.3 Toy Example: Over-parameterized CNN on FashionMNIST 50
2.4.4 Large Scale Experiments 54
2.4.5 Training and Inference Times 58
2.4.6 Adversarial Robustness 60
2.5 Conclusion 72
2.6 Appendix 73
2.6.1 Training Details 73
2.6.2 Supplementary Adversarial Results 78

3 Topology-Aware Active Learning on Graphs 83
3.1 Introduction 84
3.2 Background and Notation 87
3.2.1 Graph Laplacian Regularization 87
3.2.2 Active Learning on Graphs 89
3.2.3 Graph Curvature 95
3.3 Methods 96
3.3.1 Curvature Coreset Selection 96
3.3.2 Principled Exploration–Exploitation Balance via BFC 101
3.3.3 Localized Graph Rewiring for Label Propagation 103
3.4 Numerical Experiments 106

3.4.1 Graph Construction and Active Learning Setup 106
3.4.2 Coreset and Sequential Active Learning on Image Benchmarks 107
3.4.3 Sequential vs. Batch Active Learning 110
3.4.4 Application to PWLL 111
3.4.5 Localized Graph Rewiring 115
3.5 Conclusion and Future Work 117
3.6 Data Availability 119
3.7 Appendix 119
3.7.1 Reduction Parameter 119
3.7.2 DAC Coreset Sizes 120

4 Deep Self-Supervised Disturbance Mapping with SAR Data 122
4.1 Introduction 123
4.1.1 Our Contributions 125
4.1.2 Organization of Chapter 127
4.2 Background 127
4.2.1 Disturbance Mapping 127
4.2.2 Transformers and Self-Supervised Training 131
4.3 Methodology 133
4.3.1 Mathematical Formulation of the Model 133
4.3.2 The Disturbance Metric and Disturbance Mapping 134
4.3.3 Model Architecture, Hyperparameters, and Training 136
4.3.4 Model Inference 137
4.4 RTC-S1 Data 138

4.4.1 Training Data . . . 139
4.4.2 Data Preprocessing . . . 139
4.5 Disaster Data . . . 140
4.5.1 2024 Papua New Guinea Landslide . . . 141
4.5.2 2024 Valparaíso, Chile Wildfire . . . 142
4.5.3 2024 Bangladesh Floods . . . 144
4.6 Results . . . 145
4.6.1 Evaluation Procedure and Performance Metrics . . . 145
4.6.2 2024 Papua New Guinea Landslide . . . 148
4.6.3 2024 Valparaíso, Chile Wildfire . . . 149
4.6.4 2024 Bangladesh Floods . . . 151
4.6.5 Sensitivity to Threshold . . . 153
4.6.6 Ablation Studies . . . 154
4.7 Discussion . . . 158
4.8 Conclusion and Future Work . . . 159

5 Conclusion . . . 161

References . . . 164

LIST OF FIGURES

2.1 The graph learning layer (GLL) within a neural network pipeline compared to a standard multilayer perceptron (MLP) projection head. For the GLL, both labeled and unlabeled input images are batched into the feature extractor, which encodes the images in feature space. The GLL is a combination of two steps; it generates a similarity graph from the encoded data and then propagates the labels across the graph, giving predictions on the unlabeled data. These predictions are used in the desired loss function and gradients can flow through the GLL to update the feature extractor. By contrast, the MLP projection head pipeline does not involve any initially labeled nodes and the classification method ignores relational information in the encodings. The $^+$ indicates learnable parameters in the network; note GLL has none. [BCH24, Figure 1]. 15

2.2 Comparison between the classic structure and our proposed graph learning layer (GLL) neural network structures for image classification. Both structures consist of a feature encoder and a classifier, but we replace the multilayer perceptron (MLP) classifier - consisting of one or more linear and activation layers and a final softmax classifier - with our data-dependent GLL with no trainable parameters. We reserve a subsample of the training data - denoted *base* samples - to use as the labeled nodes for label propagation. The samples that are not from the base set are use to compute the loss. [BCH24, Figure 2] 16

2.3 The initial embedding of the two moons dataset from the random seed. The blue and red correspond to the different classes, with the starred samples corresponding to the base samples for the respective classes. [BCH24, Figure 3] 46

2.4 Visualization of effect of τ on spatial embeddings. Each row corresponds to a graph learning model with a specified value of τ or a softmax classification head. The columns correspond to the epoch of the training. With low values of τ, the model separates the data by class and resembles the original two moons shape. Increasing τ forces tighter clustering and localization. The softmax layer simply learns a linear separation with no regard to relational information. Further training with the softmax layer increases the scale, which is not seen without the axes. [BCH24, Figure 4] 49

2.5 Training loss and test accuracy of four training strategies on the **FashionMNIST** dataset with the customized feature encoder in Table 2.1. The strategies are: (1) MLP trained from scratch, (2) GLL trained from scratch (GLL-0), (3) GLL-50 (GLL after 50 epochs of MLP training), and (4) GLL-75 (GLL after 75 epochs of MLP training). For the MLP-only strategy, both the MLP and GLL losses and accuracies are recorded during training; for the remaining strategies, only the GLL classifier is used. [BCH24, Figure 5] 53

2.6 Adversarial robustness to FGSM, IFGSM, and CW attacks on the MNIST dataset. The backbone network is the same in all cases; the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 6] 68

2.7 Adversarial robustness to FGSM, IFGSM, and CW attacks on the FashionMNIST dataset. The backbone network is the same in all cases (ResNet-18); the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 7] 69

2.8 Adversarial robustness to FGSM, IFGSM, and CW attacks on the CIFAR-10 dataset. The backbone network is the same in all cases (PreActResNet-18); the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 8] 70

2.9 Comparison of accuracy vs. average distance of attack for CW attack on all three datasets. Each point represents a different value of c. We see that GLL resists the adversary from moving the target images significantly more than the standard MLP or WNLL classifier, suggesting that graph-based classification *and* precise tracking of gradients is key for adversarial robustness. [BCH24, Figure 9] 71

2.10 Visualization of FGSM, IFGSM, and CW attacks. The left, middle, and right columns show the original image, the attack, and the adversarial image, respectively. The title in the rightmost column indicates the original classification and the classification after the attack. These images were chosen to demonstrate attacks where the model was fooled by the adversary. [BCH24, Figure 10] 80

2.11 FGSM results on MNIST. The backbone network was trained in the standard way (with a softmax classifier). In the legend, MLP indicates that the FGSM attacks used the gradients from the MLP classifier (ie a linear layer and a softmax), whereas GLL used our graph learning layer. In all cases, the final classifier on the attacked images was GLL. Hence, the only difference between the results is which gradient was used for the attack. We see that the models that utilized the GLL gradients were significantly more robust to the FGSM attacks than using the standard MLP gradients. [BCH24, Figure 11] 81

2.12 Comparison of accuracy vs. average distance of attack for CW attack on both datasets. Each point represents a different value of c, where we plot additional values of c for the GLL-based models compared to Figure 2.9. We see that GLL is still superior at resisting perturbations by CW attack, even when c is orders of magnitude larger than for WNLL or MLP, and even when controlling for distance on CIFAR-10. [BCH24, Figure 12] 82

3.1 Coreset points chosen by CC and DAC at different iterations of coreset selection on the Blobs dataset. CC selects exactly one point from each of the eight clusters by the eighth iteration - the most efficient exploration of the cluster structure of the dataset possible. Moreover, they are all toward the center of each cluster (not outliers) due to the d_i terms in $\mathrm{Ric}(i, j)$. Conversely, DAC is inefficient, needing over twice as many iterations to sample from every cluster, and often sampling near the edge of the dataset. [HMW25, Figure 1] 99

3.2 Illustration of our proposed stopping condition on MNIST. Across datasets, the value of c in Algorithm 2 steadily grows until a certain "saturation" point when it starts rapidly increasing. This indicates - according to the curvature metric - that the graph topology has been sufficiently explored, and exploitative active learning may begin. The red star indicates when the online Z-score stopping condition triggers (Algorithm 3), corresponding to the first large jump in curvature values among coreset points. [HMW25, Figure 2] 100

3.3 Higher-order smoothness is imposed on the labeling function v in denser regions, while allowing greater flexibility in sparser areas. [WBT25a, Figure 3]. 104

3.4 Coreset and AL results for Curvature, DAC, and Random on several benchmarks. The top and bottom rows present results when AL begins at 50 and 100 labels, respectively. The solid line indicates the mean and shaded region indicates one standard deviation over 10 trials. Our method significantly outperforms the others, especially at the lower label rates; for reference, 50 labeled points is approximately 0.07% of the MNIST and FashionMNIST datasets and 0.08% of CIFAR-10. [HMW25, Figure 4] 109

3.5 Coreset and AL accuracy comparison between our method and DAC (with different radii), where we use each method's stopping condition. The like-colored dashed line indicates where the stopping condition is triggered (and AL begins) for each method. Across datasets, and regardless of when stopping conditions are triggered, our method significantly outperforms DAC, particularly at lower label rates. [HMW25, Figure 5] 110

3.6 Coreset and AL results for Curvature, DAC, and Random on several image benchmarks. The top and bottom rows present results when AL begins at 50 and 100 labels, respectively. We compare downstream AL performance under batch (solid line) and sequential (dashed line) uncertainty sampling AL. We report the mean over 10 trials. We exclude standard deviation shading for clarity. Not only does CC significantly outperforms the others with both AL policies, but the performance drop between sequential and batch is much less severe. This implies that CC provides a much more meaningful coverage of the graph for batch AL than DAC or Random. [HMW25, Figure 6] 112

3.7 Performance of PWLL-τ with different decay schedules. Using curvature to inform the value of τ consistently ensures effective transitioning from exploration to exploitation, and removes the need to set K beforehand in a decay schedule. The dashed line indicates the point at which our curvature-based method sets $\tau = 0$. [HMW25, Figure 7] 113

3.8 Comparison of sequential active learning performance with GBSSL variants on MNIST, FashionMNIST, and CIFAR-10. We start with 1 label per class and then run 100 AL iterations. We report mean (solid line) and standard deviation (shaded) over 100 trials. Hypergraph learning is a better classifier than Laplace learning, but manipulating the higher order weight matrix is exceedingly costly (see Tables 3.2 and 3.3). Our proposed AL Local Rewiring method strikes the best balance of accuracy and efficiency by only computing and storing the higher order terms at active learning acquisitions. [HMW25, Figure 8] 115

3.9 Comparison of reduction parameter r values for CC on CIFAR-10. r reduces the search space and helps speed up CC without sacrificing accuracy, up to a certain threshold where the search space becomes too small. [HMW25, Figure 9] 120

4.1 Visualization of the disturbance metric and how it is computed for single channel (VV) time-series. This example shows data from a landslide that occurred in Enga Province, Papua New Guinea on May 24, 2024. We use a sequence of baseline imagery $x_1, x_2, ...x_T$ (left) from before the event to predict a distribution for each pixel in the image with mean μ_{T+1} and standard deviation σ_{T+1} (center right). These parameters are used to quantify disturbance in the observed image x_{T+1} (center right, bottom) to compute one-dimensional transformer metric via the Mahalanobis distance (right), where higher values (measured in standard deviations or SD) indicate a higher likelihood of disturbance. Our methodology is able to precisely delineate the landslide disturbance extents (bottom right corner of the Disturbance Map). [HMH25, Figure 1] 126

4.2 A flow chart illustrating the transformer metric proposed in this chapter. The metric is computed for dual polarization images (see Fig. 4.1 for 1-dimensional visualization). The first arrow is where the deep-learning model is utilized to estimate the per-pixel distribution of the x_{T+1}. [HMH25, Figure 2] 133

4.3 PlanetScope (optical) imagery [PBC] of before and after the landslide in Papua New Guinea. The red polygon shows the extent of the landslide (mapped manually by the authors). [HMH25, Figure 3] 140

4.4 Landsat 8 and 9 false color imagery (bands 6, 5, 3) of before and after the fires in the Valparaíso region of Chile. The above imagery is taken from NASA Earth Observatory [NAS24]. [HMH25, Figure 4] 141

4.5 **Left**: Pre-landslide SAR imagery (VV), May 22 2024. **Middle**: Post-landslide SAR imagery (VV), June 3 2024. The landslide is seen in the bottom right corner. **Right**: Target (ground truth) damage map overlaid on ESRI Composite World Imagery [ESR24a]. The damaged area is shown in yellow. Note that the ESRI imagery is a composite and thus does not capture the recent landslide, and is used as a reference only. [HMH25, Figure 5] 142

4.6 **Left**: Pre-fire SAR imagery (VH), January 23 2024. **Middle**: Post-fire SAR imagery (VH), February 28 2024. **Right**: Target (ground truth) damage map from Copernicus overlaid on ESRI World Imagery [ESR24a]. The damaged areas are shown in yellow. Note that the ESRI world imagery is a composite and thus does not capture the recent fires, and is used as a reference only. [HMH25, Figure 6] 143

4.7 **Left**: Pre-flooding SAR imagery (VH), May 18 2024. **Middle**: Post-flooding SAR imagery (VH), July 5 2024. The flooding is seen via the reduction of the backscatter all around the Baleshwari River. **Right**: Target (ground truth) damage map from UNOSAT, overlaid on ESRI World Imagery [ESR24a]. The damaged areas are shown in yellow. Note that the ESRI imagery is a composite and thus does not capture the recent flooding, and is used as a reference only. We note that flooding continued throughout the summer, and we use data collected on July 5 as our post-event imagery to align with the damage map from UNOSAT. [HMH25, Figure 7] 144

4.8 Disturbance metrics, damage maps, and ground truth data from the landslide in Papua New Guinea. The first two columns show the model's disturbance metric before the landslide (first column) and after the landslide (second column). The third and fourth columns show the best (in terms of F_1 score) binary damage maps derived from each method's metric. The rightmost column shows the ground truth damage map. The transformer metric is the strongest, resulting in the damage map with the fewest false positives while also accurately capturing the extents of the landslide. [HMH25, Figure 8] 147

4.9 PR curves for the landslide scene. The transformer achieves the largest area under the curve (AUC). The transformer also obtains the best F_1 score (starred for each model). The disturbance delineations corresponding to the optimal τ that obtains optimal F_1 score are shown in Fig. 4.8, third and fourth columns. [HMH25, Figure 9] 148

4.10 Disturbance metrics, damage maps, and ground truth data of the fire. The first two columns show the model's disturbance metric before and after the fire. The third and fourth columns show the best (by F_1 score) binary damage maps of each method. The right column shows the ground truth damage map. The transformer metric most accurately and confidently captures the fire extents while limiting false positives. [HMH25, Figure 10] 150

4.11 PR curves for the Chilean fire scene. The transformer is the most performant model, achieving the largest area under the curve. The transformer also obtains the best F_1 score (starred for each model). The binary damage maps corresponding to these starred F_1 scores are shown in Fig. 4.10, third and fourth columns. [HMH25, Figure 11] 151

4.12 Disturbance metrics, damage maps, and ground truth data from the floods. The first two columns show the model's disturbance metric before and after the flood. The third and fourth columns show the best (by F_1 score) binary damage maps of each method. The rightmost column shows the ground truth damage map. The transformer metric is best, capturing more of the true flooding extents with fewer false positives. [HMH25, Figure 12] 152

4.13 PR curves for the Bangladesh flood scene. The transformer is the most performant model, achieving the largest area under the curve. The transformer also obtains the best F_1 score (starred for each model). The binary disturbance maps corresponding to the starred F_1 scores are shown in Fig. 4.12, third and fourth columns. [HMH25, Figure 13] 153

4.14 F_1 score vs. threshold for each method on the Papua New Guinea landslide. The threshold percentage is computed with respect to the maximum value (SD or dB) in the scene in order to compare all methods side-by-side. The transformer achieves the highest F_1 score, and does so with a wider peak compared to the other methods, which indicates less sensitivity to the choice of threshold τ. [HMH25, Figure 14] 154

4.15 F_1 score vs thresholds for the transformer metric. Effective thresholds are consistent across all three scenes and lie in the range of 4-7 standard deviations. This indicates some generality in the model and is evidence toward operational effectiveness, where the threshold τ must be chosen without knowledge of ground truth. [HMH25, Figure 15] 155

LIST OF TABLES

2.1 Network structure used in the FashionMNIST experiments (Section 2.4.3). The feature encoder is deliberately over-parameterized to create a challenging training scenario, and the model contains 1,975,424 parameters in the feature encoder and 142,346 parameters in the MLP classifier. [BCH24, Table 1] 50

2.2 Test classification error rates (%, lower is better) on FashionMNIST for the customized network in Table 2.1 under different training strategies. For the "MLP only" strategy we report both the error of the MLP classifier and that of a GLL classifier evaluated on the same encoder (GLL). For the remaining strategies, GLL is used as the classifier during training and evaluation. [BCH24, Table 2] . 52

2.3 Parameter counts by network [BCH24, Table 3] 54

2.4 CIFAR-10 test error rates (%, lower is better) comparing MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (50000 images), "20%" (10000), and "2%" (1000). Entries report mean ± one standard deviation of the test error. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Network parameter counts follow Table 2.3. [BCH24, Table 4] . 55

2.5 EMNIST test error rates (%, lower is better) comparing MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (112800 images), "20%" (22560), and "2%" (2256). "Whole" uses the full standard training set. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Entries report mean ± one standard deviation of the test error. Network parameter counts follow Table 2.3. [BCH24, Table 5] 56

2.6 EMNIST test error rates (%, lower is better) for MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (112800 images), "20%" (22560), and "2%" (2256). "Whole" uses the full standard training set. Error rates are computed on a class subindex of the EMNIST (balanced) label space: although the dataset contains 47 classes in total, we evaluate only the 10 hardest classes: 0, 1, 9, F, I, L, O, f, g, and q. Entries report mean ± one standard deviation of the test error. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Network parameter counts follow Table 2.3. [BCH24, Table 6] 57

2.7 Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is a four-layer CNN with two linear layers. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on a T4 GPU. [BCH24, Table 7] 59

2.8 Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is ResNet-18. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on a T4 GPU. [BCH24, Table 8] 60

2.9 Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is ResNet-110. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on an A100 GPU. [BCH24, Table 9] 61

2.10 Selection of adversarial results of models trained robustly and naturally with our method compared to the WNLL and MLP classifier. We report natural and adversarial error rate on FGSM, IFGSM, and CW attacks. Best error rates are in bold. For MNIST, we report $\varepsilon = 0.3$ for the FGSM and IFGSM attacks, and $\varepsilon = 0.05$ for the other datasets. We report CW results for $c = 20$. We report more complete results across a range of values for ε and c in Figures 2.6, 2.7, and 2.8. We see that GLL is significantly more robust to adversaries across datasets without sacrificing natural performance. [BCH24, Table 10] 66

3.1 Comparison of accuracy and efficiency for coreset methods after 100 labels, averaged over 10 trials. CC is consistently faster and performs better than DAC and Random. 100 labels is a 0.14% label rate for MNIST and FashionMNIST, and a 0.16% label rate for CIFAR-10. [HMW25, Table 1] 108

3.2 Accuracy and Time for 50 AL Iterations (60 total labeled points). We report means and standard deviations over 100 trials. The methods are AL Local Rewiring (AL-LR), Hypergraph (HG), Random Local Rewiring (R-LW), Laplace. [HMW25, Table 2] 116

3.3 Accuracy and Time for 100 AL Iterations (100 trials). The methods are AL Local Rewiring (AL-LR), Hypergraph (HG), Random Local Rewiring (R-LW), Laplace. [HMW25, Table 3] 117

3.4 The reduction parameter's effect on time to acquire 100 points on CIFAR-10. [HMW25, Table 4] 119

3.5 DAC Coreset Sizes Across 10 Trials. [HMW25, Table 5]. 121

4.1 Summary of Results [HMH25, Table 1] 145

4.2 Effect of input (I) and patch sizes (P) on transformer performance. [HMH25, Table 2] 155

4.3 Effect of input size on RNN performance. [HMH25, Table 3] 156
4.4 Effect of width of feedforward layers (FF), number of layers (L), and total number of parameters (P) on transformer performance. [HMH25, Table 4] 157
4.5 Effect of learning rate on transformer performance. [HMH25, Table 5] 157

ACKNOWLEDGMENTS

I've always been really excited to write the Acknowledgments section of my thesis. Maybe it was the opportunity to inject a little personal prose into an otherwise factual, mathematical collection of writing. Maybe it would feel cathartic (it does). Most of all, I think I've been looking forward to these few pages because this dissertation marks the end of a chapter of my life. And I'm very lucky to have a lot of people to thank - not just from the last five years, but twenty-seven years.

People who have electrified and animated my life, given me timely wisdom or encouragement or a road map forward, sacrificed for me, challenged me, supported me, stoked my passions and shared them, taught me the coolest stuff I know, fed me, answered my incessant questions, watched a game with me, gotten extremely worked up over the outcome of a game with me, laughed with me, laughed at me, gave me a reason to laugh, suffered with me, given me a place to lay my head at night, joined me on an adventure, or just been fun to be around. My colleagues, my teammates, my coaches, my teachers, my friends, my family. By being who you are, you became a part of my village; I am the sum of those people.

Professional Acknowledgments

Thank you to my advisor, Andrea Bertozzi. Andrea welcomed me into her group at the end of my first year and introduced me to the world of graph-based learning during my first summer research project with her. Her group's work inspired and laid the foundation for my own. She connected me with key collaborators throughout academia and national labs - leading to many fruitful projects and an internship at JPL - and supported me through many important milestones during my Ph.D. Thank you for your time, guidance, and stories.

Thank you to Jeff Calder, who I have had the privilege to collaborate with on two papers. I once remarked to a friend that every time I meet with Jeff, I learn something new. Jeff also created and maintains the `graphlearning` Python package, which has been a genuinely invaluable tool for my research.

I also want to express gratitude to the remaining members of my committee, Deanna Needell and Hayden Schaeffer, for serving on the committee and being an important part of my graduate education. I had the pleasure of taking two courses with Deanna, and she kindly invited me to her group meetings as I was exploring research topics, where I learned how to share a "rose-bud-thorn". I also want to especially thank her for being so accommodating when I was scheduling my ATC and my Defense (this is arguably the hardest part of doing a Ph.D.). I met Hayden a little later on in graduate school, and always enjoyed our discussions about operator learning during office hours and his humor.

Another UCLA professor I want to acknowledge is Prof. Jonathan Kao in the ECE Department, who taught two graduate classes on deep learning that I always recommend to new applied math students. My understanding of the field, my research, and my coding have all been heavily informed by his courses.

Thank you to my JPL mentors, Charlie Marshak and Al Handwerger. My internship was a formative experience and changed how I think about my interests and career. Charlie and Al instilled a lot of confidence in me - they gave me autonomy to lead the project, but were ever-present to provide help when I needed it. They were also super fun to work with!

In addition to those already mentioned, I was fortunate to work on projects with many talented people: Joshua Enwright, Jason Brown, Bohan Chen, Adrien Weihs, Jocelyn Chanussot, and Jack Mauro. Thank you for your hard work, collaboration, and for being a part of my development as a researcher. Specifically, I want to give credit to the people who worked with me on the papers appearing in this thesis.

Chapter Acknowledgments

Chapter 2 is a version of [BCH24], which is currently in revision at the Journal of Machine Learning Research and is a joint work with Jason Brown, Bohan Chen, Jeff Calder, and Andrea L. Bertozzi. Jeff proposed the idea and contributed the mathematical derivations. Jason, Bohan, and I contributed the implementation and experiments, and wrote the paper

as co-first authors. Jeff and Andrea advised the project and provided feedback throughout. This chapter expands significantly on work I did previously in [EHC23], which was a joint work with Joshua Enwright, Jeff, and Andrea. During this project, I was supported in part by NSF research training grant DGE-1829071.

Chapter 3 is a version of [HMW25], currently under review at Pattern Recognition. I was the lead author and the project is joint work with Jack Mauro, Adrien Weihs, and Andrea L. Bertozzi. Jack, Adrien, and I collaborated to propose the initial ideas. I developed and implemented the curvature-based methods. Jack and I contributed the implementation of the hypergraph methods, and ran the curvature and hypergraph experiments. Adrien provided the initial hypergraph codebase and gave helpful feedback on the methods and experiments. All three of us wrote the paper. Andrea supervised the project and provided valuable feedback on data, experiments, and writing. Thank you also to James Chapman for his helpful conversations about coresets early on in this project. This chapter also used computational and storage services associated with the Hoffman2 Cluster which is operated by the UCLA Office of Advanced Research Computing's Research Technology Group.

Chapter 4 is a version of [HMH25], published in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS) and adapted here under a CC BY 4.0 license. This is a product of my internship at NASA Jet Propulsion Lab (JPL), mentored by Charlie Marshak and Al Handwerger. Charlie and Al proposed the internship project and provided the data. I led the ideation, developed and implemented the model, ran the experiments, and wrote the paper as first author under their supervision. My work was supported in part by the 2024 UCLA JIFRESSE Summer Internship Program (JSIP) and in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). Thank you to the OPERA Project team, who provided early feedback and suggestions about this work.

Funding Acknowledgments

I want to acknowledge support from the National Geospatial-Intelligence Agency (Award

No. HM0476-21-1-0003), the National Science Foundation Research Traineeship (Grant No. DGE-1829071), and the National Science Foundation Graduate Research Fellowship Program (Grant No. DGE-2034835), which provided funding (either in part or in full) for all of the work presented in this thesis. Without it, my research would have been considerably more difficult or even impossible to complete. Fund science, and elect people who will fund science.

Additionally, I acknowledge support for my co-authors. In Chapter 2, JB and ALB were supported by Simons Math + X award 510776. ALB and BC were supported by NSF grants DMS-2027277 and DMS-2318817. BC was supported by the UC-National Lab In-Residence Graduate Fellowship Grant L21GF3606. JC was supported by NSF-CCF:2212318, the Alfred P. Sloan Foundation, and an Albert and Dorothy Marden Professorship. For Chapter 3, ALB was supported by NSF grants DMS-2152717 and DMS-2318817 and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, USA under award DE-SC0025589.

Personal Acknowledgments

I would not be here without some incredibly influential teachers and mentors I had throughout my education. Thank you to Mark Headlee (5th grade), for being the first person who told me I was talented at math, and pushed me to do well in it; Tonya VanOrden (7th and 8th grade science), for getting me super excited about science, and channeling my stubbornness and curiosity in the right direction (she once noticed I was frustrated with math we were learning, and told me "math is the language of science," to which I responded "it's also the language of my annoyance!" Safe to say she put up with a lot.); and Kim Schjelderup (AP Calculus AB and BC, AP Microeconomics, AP Macroeconomics), for running the most organized and tightest ship in the history of AP Calculus, and preparing me incredibly well for my first year of college.

At Tufts, I was planning on studying engineering. After my 2nd year, I wasn't enjoying the coursework as much as I wanted to; I was feeling stuck and unsure of what to pursue. Realizing the classes that were most exciting to me were in math, I signed up to take Real

Analysis to see if a math major would suit me better. I was incredibly fortunate that the instructor was Prof. Loring Tu. Prof. Tu delivers the best, clearest lectures I've ever had the pleasure of attending, and dedicates a tremendous amount of time to his students outside of class. Real Analysis is still my favorite class I've ever taken (not to mention Real Analysis II and Differential Geometry, also with Prof. Tu), and set me on a path toward a math major and Ph.D.

It was around this same time I reached out to Prof. James Murphy about doing a summer research project with him. James was incredibly welcoming and accommodating, writing a recommendation letter on days' notice to secure funding for me for the summer. The project was my first introduction to math research and thoroughly stoked my interest in machine learning. James was also the person that encouraged me to pursue a Ph.D., which, at the time, was barely on my radar; he believed in me and gave me the push exactly when I needed it. Without any doubt, I wouldn't have done a Ph.D. without James' mentorship and advice.

Next, I want to acknowledge people from my personal life, which has been filled with so many wonderful, unique, and genuine individuals.

I got to grow up with Matt, Josh, Roberto, Ethan, Chris, and Nathan. I'm so grateful to have gone from 3rd through 12th grade with them, and still have them in my life now. One of the special things about this group was I never had to dim any part of myself to belong. We all took the same classes, studied, and played sports until the sun went down (sometimes long after), together. Lots of video games too. And fantasy football. One day I'll win that stupid league.

I started rowing when I was 14 (my mom made me sign up for it - thanks Mom) at the Mount Baker Rowing and Sailing Center and it quickly became a foundational part of my young adulthood. It is a brutally amazing sport. You have to be a little insane to enjoy it, but the benefit is that you get to meet a lot of like-minded, mildly crazy people. Some of them even become your friends. In high school, Nolan and I were fast friends - equally obsessed with our rowing technique, the latest financial news, or our most recent regatta dog

sighting.

Rowing was also a large factor in my college decision. There, I met some of my closest friends: Henry, Eoghan, Alex, Tara, John F., John G., Aidan, Jack B., Rachel, Mitchell, Nina, and Sophie. I miss our endless breakfast and dinner conversations, sleep-deprived problem set sessions, hanging out on a porch as the sun set, and invariably making our way to Late Night, but I equally look forward to the next bike ride or run together, sharing another BLL, and attending (in some cases, already attended) your weddings! Henry, thank you for your friendship in college and beyond. I've always felt like you were built in a lab to be my friend. I can - and will - talk to you about anything. I also want to thank Coach George. He's a gifted communicator, motivator, and the best coach I've had in any sport. I've never improved more at anything the way I did rowing under George, and he pushed the team so hard that a Ph.D. seemed pretty manageable by comparison. Above all, he's an awesome person.

Thank you to all my rowing teammates and coaches. Even if you aren't listed here, thank you for all the times you were with me before the sun came up, cold, wet, and exhausted, but making the boat go fast.

On a rare occasion, I branched out and met some non-rowing friends. Tyler and I bonded over our love of hoops, getting ice cream at CB scoops, and the many interesting questions and answers we encountered during ES-2 office hours and exams. Dan and Taylor (in less formal settings, DanTaylor) were my engineering classmates and lab partners that turned into my close friends and fellow wine enjoyers. I'm so lucky they followed me out to southern California for a while (and convinced me to invest in a good PC), and can't wait for our next chance to drop a Vivino review together.

At UCLA, I got to share in the many twists and turns of graduate school with a lovely group of friends, including Megan, Advika, Emmy, Eric, Olha, Thomas, Rushil, Tanvi, Jas, Sylvie, Shiraz, and Jack. I won't miss studying for quals but I will miss doing it with you! Thank you for your camaraderie, exploring LA with me, and finding any reason to throw

a party (typically with an unusual theme). I am sad at the prospect of us dispersing after graduating, but so thankful to have you all as lifelong friends.

Thank you to my partner Lani. Your daily love and support mean the world to me, be it perfectly-timed Sidecar Donuts and Perfect Latte pick-me-ups or your uplifting, reassuring words and presence. You are endlessly thoughtful, funny, and beautiful in every way. I am so grateful to have you in my life. Thank you to the rest of the Owen family - Maile, Colin, Andrea, and Ray - for welcoming me in so warmly and so generously.

My family means everything to me. They are the most caring, supportive, loving, and fun collection of people I could've asked for, and I love them a lot. To my aunt Leslie and uncle Doug (a.k.a. Bubba), thank you for your presence, your steadiness, and all the celebratory sweet treats you provided over the years. I miss seeing you in Portland but hope I can visit you in North Carolina soon. To my aunt Hare, uncle Doug (a.k.a. Mux), and cousin Miranda, thank you for adding so much light, kindness, and enthusiasm to my life. The days we spend together around Christmas are my favorite days of the year, every year.

To my grandmother Susie (a.k.a. Mim), thank you for being such a consistent part of my life. You give me so much love and comfort. I love spending time with you, whether it's late summer at the Ranch and we get to watch the aspen trees change color, or Thanksgiving in Seattle as Dad forces us to go for yet another walk. Wherever it is, I know there will always be plenty of coffee and wine for us to share together.

To my sister Rhys, I will always adore your thoughtfulness, your spirit, and your limitless humor. I'm so proud of you and can't wait to watch you continue to grow. I love that we laugh at all of the same things - usually, at how bad our NBA teams are doing, or re-watching our favorite episode of Game of Zones or Gravity Falls. "Awkward Sibling Hug? Pat. Pat."

Mom and Dad, my dissertation is dedicated to you. I wouldn't be here without your infinite patience, kindness, encouragement, and love, not to mention your impressive ability to keep 16-year-old Harris fed. I look forward to every second we get to spend together.

Dad, thank you for being such a dedicated father while also tirelessly working to support us all this time. I can't comprehend how you managed to do so well at both. I try to take your unique blend of intellectual curiosity and nonstop joking with me wherever I go. Every day out on a trail or riding our bikes together is a perfect day to me (especially when followed by grilled meats and wine). You are such an incredible role model for me. I hope someday I can be as great a father as you were to me, including driving to the airport baggage claim on a Saturday morning. Mom, thank you for being my biggest supporter and advocate, every single day for the last 10,000 days. You sacrificed so much to be constantly available for whatever I needed and your selflessness as a mother has always made me feel secure and celebrated as I reached milestones big and small throughout my life. You also hold the permanent position of my favorite person to watch a Seahawks game with. Thank you for your endless affection, for your comfort when things got hard, and for nurturing an irrepressibly curious and precocious little kid into the person I am today. And the many, many Triple Chocolate Cakes. You're the best mom ever.

VITA

2021	B.S. in Mathematics, B.S. in Mechanical Engineering, *Summa Cum Laude*, Tufts University
2021	Ralph S. Kaye Memorial Prize (Top Mathematics Student), Tufts University
2021	Frederick Melvin Ellis Prize, for "marked athletic versatility, academic achievement, and effective leadership", Tufts University
06/21-08/21	Graduate Data Science Intern, The MITRE Corporation
2021-2022	NSF MENTOR Fellowship
2022-2023	Teaching Assistant, UCLA Department of Mathematics
2022-Present	Graduate Student Researcher, UCLA Department of Mathematics
2023	M.A. in Mathematics, UCLA
2023-2026	NSF Graduate Research Fellowship
07/24-09/24	Machine Learning Research Intern, NASA Jet Propulsion Lab

PUBLICATIONS

H. Hardiman-Mostow, J. Mauro, A. Weihs, A.L. Bertozzi. "Topology-Aware Active Learning on Graphs." Under Review, *Pattern Recognition.* arXiv:2510.25892, 2025.

H. Hardiman-Mostow, C. Marshak, A. Handwerger. "Deep Self-Supervised Global Disturbance Mapping with Sentinel-1 OPERA RTC Synthetic Aperture Radar." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2025.

J. Brown*, B. Chen*, H. Hardiman-Mostow*, J. Calder, and A.L. Bertozzi. "GLL: A Differentiable Graph Learning Layer for Neural Networks." In Revision, *Journal of Machine Learning Research*. arXiv:2412.08016, 2024.

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

J. Enwright*, H. Hardiman-Mostow*, J. Calder, and A.L. Bertozzi. "Deep Semi-supervised Label Propagation for SAR Image Classification." *Algorithms for Synthetic Aperture Radar Imagery XXX*, vol. 12520, pp. 158-170. SPIE, 2023.

* indicates equal contribution

CHAPTER 1

Introduction

1.1 Overview of the Dissertation

Supervised machine learning algorithms typically require vast amounts of labeled training data. However, in many real-world applications - such as medical imaging, remote sensing, or scientific data analysis - labels are expensive or impractical to obtain at scale, while unlabeled data is often abundant. To circumvent the need for large training sets, some learning strategies seek to reduce or eliminate reliance on labels without sacrificing performance. When a small training set is available, *semi-supervised learning* supplements limited ground truth information with the structure inherent in the dataset. When labels are completely unavailable, *self-supervised learning* extracts a training signal solely from the data itself. This dissertation develops methods in both paradigms.

We begin with semi-supervised learning. One way to represent the structure of a dataset is to compute a *similarity graph* that encodes pairwise relationships between data points. *Graph-based semi-supervised learning* (GBSSL) lifts the learning problem onto the similarity graph to exploit its geometry, leveraging both labeled and unlabeled data to inform the task at hand. Beginning with the seminal work by Zhu et al. [ZGL03], GBSSL has since proliferated [BN04, BMN04, BDL06, ZB11, WTT13, HLS05, LHW13, CCT20, CS20a, SOZ17] and exhibits strong performance in diverse applications such as image processing [ITA19, SAS22], remote sensing [MM18, BOC23, EHC23], data fusion [ICB20], adversarial robustness [WO21], few-shot learning [LLP19, HZZ21], and uncertainty quantification [BLS18]. Even in

the era of deep learning, GBSSL remains an active area of research due to its compelling blend of empirical results, theoretical tractability, and interpretability (see e.g. [WBT25a,WBT25b] and references therein).

GBSSL consists of two distinct but important steps: *imposing* structure by building a similarity graph, and *leveraging* the structure using a learning algorithm. We address both elements: the former in Chapter 2, and the latter in Chapter 3.

Since GBSSL operates on a similarity graph constructed over the data, the performance of the algorithm depends critically on the representation of that data. Often, basic metrics (such as Euclidean distance) do not provide a meaningful notion of similarity between raw data points. For example, the classification of an image is preserved under transformations (e.g. rotation, translation, noise) that can cause drastic changes in distances between two similar samples. Instead, it is common to preprocess the data by embedding points into a latent space where pairwise distances can better reflect similarity *before* constructing the graph. Prior works have done this with neural networks [CCT20, CE22, MMS22, CCT23, BOC23, SAS22, ITA19, WO21]. However, network training is not end-to-end; the gradient of the loss function with respect to graph construction and graph learning is approximated, or the two processes are decoupled altogether. This motivates a framework that fuses these components into a unified algorithm.

In Chapter 2, we induce *structure from learning* by integrating graph learning classifiers into an end-to-end neural network, enabling the model to directly learn latent representations that yield effective graphs for GBSSL. We derive backpropagation equations, via the adjoint method, for a general family of graph learning algorithms in a gradient-based learning environment. This allows us to incorporate similarity graph construction and graph-based learning into a neural network as a fully differentiable layer, which we call the Graph Learning Layer (GLL). The GLL replaces the standard projection head and softmax activation function, which do not utilize any relational information between samples that may inform classification. In experiments, models equipped with a GLL demonstrate smooth label tran-

sitions across data, improved generalization and robustness to adversarial attacks, and more stable training dynamics compared to both the standard softmax-based approach and prior work that approximates gradients [WO21], with particular gains at low label rates.

We now turn to harnessing the information encoded in the similarity graph. In semi-supervised learning, we assume a small labeled set; there is also typically a high cost to generating new labels in practice (e.g., time spent by a human expert). In this context, the *choice* of which data points to label can drastically impact classifier performance and affect real-world resource allocation. Graph-based *active learning* (AL) [Set09] seeks to address this by carefully selecting a limited number of unlabeled points to query for labels from an expert "human-in-the-loop" based on the graph's geometry and the state of the current classifier. Methods that only use the former are known as *coreset* methods [CCT23], which are often advantageous when the classifier is poorly calibrated at early stages. When combined, GBSSL and AL can beat state-of-the-art supervised neural network methods with a fraction of the training data [CCT23], and in some cases *orders of magnitude* less training data [CMB25].

A key challenge in AL is the *exploration-exploitation trade-off*: queries may be used to explore unlabeled regions of the graph, or exploit current knowledge to refine predictions. In Chapter 3, we *learn from structure* by proposing a graph-topological approach to AL that directly targets this trade-off under scarce label budgets. To guide exploration, we introduce a coreset algorithm based on Balanced Forman Curvature (BFC), which selects representative initial labels that reflect the graph's cluster structure. Our method includes a data-driven stopping criterion that signals when the graph has been sufficiently explored. We further apply BFC to dynamically trigger the transition from exploration to exploitation within active learning routines, replacing hand-tuned heuristics. To improve exploitation, we introduce a localized graph rewiring strategy that efficiently incorporates multiscale information around queried nodes, enhancing label propagation while preserving sparsity in computations. Experiments on benchmark image classification tasks illustrate that our methods

surpass established baselines, achieving stronger performance with smaller label budgets in both sequential and batch active learning regimes, while offering significant computational savings.

In Chapter 4, we shift our focus to a fully label-free setting and propose a *self-supervised* approach to near-global disturbance mapping using satellite remote sensing data. Mapping land surface disturbances supports disaster response, resource and ecosystem management, and climate adaptation efforts. Earth-orbiting satellites continuously generate massive amounts of imagery, providing a critical data source for detecting such disturbances. However, the volume and complexity of time-series remote sensing data often render large-scale training sets prohibitively expensive. This motivates the self-supervised approach we develop in this Chapter.

Among remote sensing techniques, synthetic aperture radar (SAR) is an invaluable tool for disturbance mapping, as it provides consistent time-series images of the ground regardless of weather or illumination conditions. Despite SAR's potential, it has been underutilized in applications; processing SAR data to an analysis-ready format requires expertise and significant compute resources, particularly for large-scale global analysis. In October 2023, NASA's Observational Products for End-Users from Remote Sensing Analysis (OPERA) project released the near-global Radiometric Terrain Corrected SAR dataset from Sentinel-1 (RTC-S1 dataset), providing publicly available, analysis-ready SAR imagery. In this Chapter, we use this new dataset to systematically analyze land surface disturbances. We train a self-supervised vision transformer to estimate a per-pixel distribution from the set of baseline imagery and assess disturbances when there is significant deviation from the modeled distribution. To test our model's capability and generality, we evaluate it on three natural disasters - which represent high-intensity, abrupt disturbances - from geographically diverse regions in the world. Across events, our approach yields high-quality disturbance identification: F_1 scores above 0.6 and Areas Under the Precision-Recall Curve over 0.65, exceeding the metrics of previous SAR disturbance methods. Our findings suggest that a

self-supervised vision transformer is well-suited for global disturbance mapping and can be a valuable tool for operational, near-global disturbance monitoring, particularly when labeled data is unavailable.

1.2 Graph-Based Semi-Supervised Learning

As Chapters 2 and Chapters 3 both concern GBSSL, we consolidate some shared mathematical background in this Section. We introduce the GBSSL problem setting, key concepts, and notable work from over two decades of development, intended to be accessible to researchers new to the area. While not exhaustive, we provide sufficient background for the scope of this dissertation. Nevertheless, all Chapters are mathematically self-contained and will introduce relevant background and notation as needed.

1.2.1 Problem Formulation and Notation

Consider a set of n feature vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} =: \mathcal{X}$, where each $\mathbf{x}_i \in \mathbb{R}^d$. These feature vectors may be the raw data from the task at hand (e.g., images), but more often are latent representations of the data derived from some preprocessing step, such as an unsupervised neural network [MMS22]. Suppose we are given m labeled data points with labels y_i, while the remaining $n - m$ points are unlabeled. Without loss of generality, it will be helpful to partition the data into an unlabeled set $\mathcal{U} := \{\mathbf{x}_1, \ldots, \mathbf{x}_{n-m}\}$ and labeled set $\mathcal{L} := \{\mathbf{x}_{n-m+1}, \ldots, \mathbf{x}_n\}$. We assume each point $\mathbf{x}_i$ belongs to one of c classes. In binary classification ($c = 2$) labels y_i take values in $\{0, 1\}$. In multiclass classification ($c > 2$), the labels are represented by one-hot vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_c\}$, where $\mathbf{e}_i$ is the i^{th} standard basis vector and the label for the i^{th} class. In semi-supervised learning, the labeled set is typically small compared to the unlabeled set (that is, $m \ll n - m$) and the task is to infer the labels of $\mathcal{U}$ using $\mathcal{L}$.

1.2.2 Graph Construction

In GBSSL, we construct an undirected *similarity graph* $G(\mathcal{X}, W)$, treating the data $\mathcal{X}$ as nodes and assigning nonnegative, symmetric edge weights $w_{ij} = w_{ji}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$. We call the $n \times n$ matrix $W = (w_{ij})_{i,j=1}^n$ the similarity matrix, where w_{ij} encodes how similar $\mathbf{x}_i$ and $\mathbf{x}_j$ are: large values indicate high similarity, while values close or equal to zero indicate little or no similarity. Generically, we compute W as $w_{ij} = \eta\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\varepsilon}\right)$ where η is a non-increasing kernel function and $\varepsilon > 0$ is a scale parameter. The distance metric is commonly angular distance (derived from cosine similarity), which is often more useful than Euclidean distance in high dimensions. The specific formulation we use in this thesis is a variant of the Gaussian kernel,

$$w_{ij} = \exp\left(\frac{-4\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\varepsilon_k(\mathbf{x}_i)\varepsilon_k(\mathbf{x}_j)}\right),$$

where $\varepsilon_k(\mathbf{x}_i)$ is the distance from $\mathbf{x}_i$ to its k^{th} nearest neighbor. This enables the weights to "self-tune" to areas of density or sparsity in the graph.

Computationally, it is usually prohibitive to compute, store, and manipulate a fully dense W for large datasets. Instead, this thesis will build a k-Nearest Neighbors graph, where we only compute w_{ij} if $\mathbf{x}_j$ is among the k nearest neighbors of $\mathbf{x}_i$, and set $w_{ij} = 0$ otherwise. Taking $k \ll n$ ensures sparsity in W, significantly lowering computational costs. For example, experiments in Chapter 3 will use $k = 25$ or 50, while $n = 70000$. An exact k-NN search in high dimensions is challenging, so we employ a fast approximate algorithm called Annoy [Ber24]. k-NN can introduce asymmetry in W - $\mathbf{x}_i$ may not be a top k neighbor of $\mathbf{x}_j$, even if the reverse is true - so it is standard to replace the resulting W with $(W+W^\top)/2$.

1.2.3 Laplace Learning

Equipped with a similarity graph, we can now return to the semi-supervised learning problem at hand. For simplicity, let us consider binary classification. Our goal is to generate a

labeling function $\mathbf{u} : \mathcal{X} \to \mathbb{R}$ to infer labels on $\mathcal{U}$. While it is common to write $\mathbf{u}$ as a function on the graph, where $\mathbf{u}(\mathbf{x}_i)$ is the prediction at $\mathbf{x}_i$, it can also be interpreted as a vector in $\mathbb{R}^n$, where each scalar entry u_i is the label prediction for $\mathbf{x}_i$. Similarly, we write $(y_{n-m+1}, \dots, y_n)^\top =: \mathbf{y} \in \mathbb{R}^m$ as the vector of known labels (recalling our convention that the last m data points are labeled).

Clearly, extending the labels from $\mathcal{L}$ to $\mathcal{U}$ is ill-posed; there are infinitely many ways to do this. GBSSL regularizes the problem with the so-called Semi-Supervised Smoothness Assumption, which posits that $\mathbf{u}$ should vary smoothly over the graph; similar nodes should be likely to share labels. We can formalize this notion with the graph Dirichlet energy functional

$$E(\mathbf{u}) := \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2.$$

Minimizing this energy encourages u_i and u_j to be close whenever w_{ij} is large, as desired. Writing the (diagonal) degree matrix D of the graph as $D_{ii} = \sum_{j=1}^{n} w_{ij}$, we can define the (unnormalized) *graph Laplacian* as $L = D - W$, and rewrite the energy as $E(\mathbf{u}) = \mathbf{u}^\top L \mathbf{u}$.

We are now prepared to introduce *Laplace learning* [ZGL03], the seminal GBSSL method and a key algorithm used in this thesis. Laplace learning minimizes the Dirichlet energy subject to the constraint that the known labels are retained:

$$\begin{aligned} \min \quad & \mathbf{u}^\top L \mathbf{u} \\ \text{s.t.} \quad & \mathbf{u} = \mathbf{y} \text{ on } \mathcal{L}. \end{aligned} \tag{1.2.1}$$

This can be interpreted as finding the smoothest labeling function $\mathbf{u}$ that agrees with the given labels. Often, (1.2.1) is instead written as

$$\begin{aligned} L\mathbf{u} &= 0 \text{ on } \mathcal{U} \\ \mathbf{u} &= \mathbf{y} \text{ on } \mathcal{L}. \end{aligned} \tag{1.2.2}$$

A brief word on notation. By "on $\mathcal{L}$", we mean the rows corresponding to the labeled points - by our convention, this would mean the last m rows of $\mathbf{u}$. To be more explicit, we often use the subscript $\mathcal{U}$ or $\mathcal{L}$ to indicate restriction to rows or columns corresponding to labeled or unlabeled data (slightly abusing notation by treating sets of points as their index sets). For example, the hard label constraint in (1.2.2) could be written as $\mathbf{u}_{\mathcal{L}} = \mathbf{y}$, and we can rewrite the first equation in (1.2.2) explicitly as a linear algebra problem,

$$L_{\mathcal{U}\mathcal{U}}\mathbf{u}_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{L}}\mathbf{y}. \tag{1.2.3}$$

While L is positive semi-definite, $L_{\mathcal{U}\mathcal{U}}$ is positive definite provided that there is at least one labeled node in every connected component of the graph, so the solution to (1.2.3) exists and is unique. Label predictions can be obtained by thresholding the solution $\mathbf{u}_{\mathcal{U}}$ at 0.5 - those below the threshold are predicted to belong to Class 0, and those above to Class 1.

A derivation of (1.2.3) from (1.2.1) is often skipped, but we include it here for completeness. Recalling the ordering convention for unlabeled and labeled data we adopted earlier, we can write L using block notation as follows:

$$L = \begin{pmatrix} L_{\mathcal{U}\mathcal{U}} & L_{\mathcal{U}\mathcal{L}} \\ L_{\mathcal{L}\mathcal{U}} & L_{\mathcal{L}\mathcal{L}} \end{pmatrix},$$

and similarly $\mathbf{u} = (\mathbf{u}_{\mathcal{U}}, \mathbf{u}_{\mathcal{L}})^\top$. Next, we expand the quadratic form:

$$\mathbf{u}^\top L\mathbf{u} = \mathbf{u}_{\mathcal{U}}^\top L_{\mathcal{U}\mathcal{U}}\mathbf{u}_{\mathcal{U}} + 2\,\mathbf{u}_{\mathcal{U}}^\top L_{\mathcal{U}\mathcal{L}}\mathbf{u}_{\mathcal{L}} + \mathbf{u}_{\mathcal{L}}^\top L_{\mathcal{L}\mathcal{L}}\mathbf{u}_{\mathcal{L}},$$

where the factor of 2 in the middle term is from the symmetry $L_{\mathcal{L}\mathcal{U}} = L_{\mathcal{U}\mathcal{L}}^\top$. Recall that by the constraint in (1.2.1), we only control the predictions on the unlabeled nodes $\mathbf{u}_{\mathcal{U}}$. Hence, we take the gradient with respect to $\mathbf{u}_{\mathcal{U}}$ and set it to zero:

$$\frac{\partial}{\partial \mathbf{u}_{\mathcal{U}}}\mathbf{u}^\top L\mathbf{u} = 2L_{\mathcal{U}\mathcal{U}}\mathbf{u}_{\mathcal{U}} + 2L_{\mathcal{U}\mathcal{L}}\mathbf{u}_{\mathcal{L}} = \mathbf{0}.$$

This yields the optimality condition

$$L_{\mathcal{U}\mathcal{U}}\mathbf{u}_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{L}}\mathbf{u}_{\mathcal{L}},$$

as desired, since we use the hard label constraints $\mathbf{u}_{\mathcal{L}} = \mathbf{y}$.

For the multiclass setting, we simply solve (1.2.3) c times - once for each of the c classes - and make predictions by taking an argmax over the classes. More precisely, we solve

$$L_{\mathcal{U}\mathcal{U}}U_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{L}}Y$$

where now $U : \mathcal{X} \to \mathbb{R}^c$ (alternatively, U is a matrix in $\mathbb{R}^{n\times c}$) and $Y \in \mathbb{R}^{m\times c}$ is a matrix whose rows are the labels $\{\mathbf{y}_i\}_{\mathbf{x}_i\in\mathcal{L}}$ and we use one-hot label vectors, $\mathbf{y}_i \in \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_c\}$. To be clear, in binary classification, the individual labels y_i were scalars and the entire set of labels $\mathbf{y}$ was a vector; in multiclass classification, the individual labels $\mathbf{y}_i$ are vectors, and the entire set of labels Y is a matrix. We keep the condition $U_{\mathcal{L}} = Y$. This is a system of c independent equations, and each entry U_{ij} indicates the confidence that $\mathbf{x}_i$ belongs to class j. This is known as a "one-versus-rest" approach. The label prediction for $\mathbf{x}_i$, $\hat{y}(\mathbf{x}_i)$, can then be made by taking the class with the highest value:

$$\hat{y}(\mathbf{x}_i) = \operatorname*{argmax}_{j\in\{1,\ldots,c\}} U_{ij}.$$

1.2.4 Additional Background

The Laplacian L encodes rich information about the graph topology. For instance, L always has an eigenvalue of 0 whose multiplicity equals the number of connected components, and the corresponding eigenspace is spanned by the indicator vectors of those components [Lux07, Proposition 2]. These spectral properties are central to many graph-based learning methods such as spectral clustering [NJW01], and have been studied extensively in the context of statistical consistence and convergence [GS18, HHO22]. In semi-supervised

learning, Laplacian-based regularization with $\mathbf{u}^\top L\mathbf{u}$ penalizes the first-order variation of the labeling function over the graph [ST19]. It serves as a discrete analog of the squared gradient norm, encouraging smoothness between neighboring nodes.

Proposed in 2003, Laplace learning [ZGL03], also known as *label propagation*, led to a surge of related techniques [BN04, BMN04, BDL06, ZB11, WTT13, HLS05, LHW13, GMB14, CS20a, SOZ17]. In contrast to modern deep learning, many GBSSL methods enjoy theoretical performance guarantees, derived using variational analysis, random walks, harmonic functions, and partial differential equations [Cal19,CG22,CS20a,Cal18,ST19,DSS20,GGH20, HAL07,HAL05,CE22,CST23]. Moreover, GBSSL is a class of *transductive* machine learning algorithms, where labeled and unlabeled data are considered together for the learning task. Most deep learning methods are *inductive*, learning a set of rules from the labeled data to then apply to the unlabeled data. The transductive paradigm is often preferable when labels are scarce [Joa99, CMB25, CCT23]. Graph learning also differs from deep learning in that it is a data-dependent model without any trainable parameters.

When the label rate is extremely low (e.g., the number of labels are fixed while the amount of unlabeled data tends to infinity), Laplace learning can degenerate and suffer from a "spiking" phenomenon around labeled points, while predictions are roughly constant [CCT20,ECR16,NSZ09]. Several works have analyzed this behavior, leading to new methods for very low label rate regimes such as reweighted Laplace learning [SOZ17, CS20a, MC23], p-Laplace methods [ZB11,ECR16,FCL22,ST19], and Poisson learning [CCT20]. While these alternatives offer improvements for very low label rates, the work in this thesis operates in regimes where the label rate is sufficient for Laplace learning to remain well-behaved, and we will use it as our primary GBSSL algorithm. Some recent work has also focused on graph total variation problems using graph cuts [BF16, MBY17, MBC18].

Another related, but distinct, area of work in machine learning on graphs is graph neural networks, or GNNs [KW16, DBV16, WPC20, KBM23]. GNNs typically operate on an *a priori* fixed graph of relevant meta-data to the task (such as co-authorship in a citation

graph), and train a neural network for various learning tasks. Conversely, we assume no given graph structure, instead constructing one using feature similarity, and, as previously noted, GBSSL methods have no trainable parameters. Some works such as graph attention networks [VCC17] learn how to assign weights to a graph, but the graph adjacency structure is still fixed through masking in the attention mechanism. However, in Chapter 3, we will employ a formulation of graph curvature proposed in the GNN community [TGC22] to inform coreset selection and AL routines in similarity graph-based semi-supervised learning. The intersection of ideas from GBSSL and GNNs is an underexplored area and represents an important line of future work.

CHAPTER 2

GLL: A Differentiable Graph Learning Layer for Neural Networks

This Chapter is an adaption of [BCH24], which is a joint work with Jason Brown, Bohan Chen, Jeff Calder, and Andrea L. Bertozzi. Jeff proposed the initial idea and contributed the derivations. Jason, Bohan, and I implemented the method, ran the experiments, and wrote the paper as first authors under Jeff and Andrea's supervision.

This Chapter studies the first component of GBSSL: encoding structure by building effective similarity graphs. To this end, neural networks are often used in tandem with graph-based semi-supervised learning, where the former learns an embedding to a suitable latent space and the latter handles similarity graph construction and classification. However, the graph learning step is typically excluded from neural network training, and existing attempts to include it rely on approximate gradients. In this Chapter, we derive and implement a novel Graph Learning Layer (GLL) that handles forward and backward propagation for sparse graph construction and a general family of graph learning algorithms, allowing for precise integration into any neural network. Experiments demonstrate that GLL boosts generalization and strengthens adversarial robustness compared to models trained with standard projection heads or graph-based heads that only approximate the backward gradients of graph learning. These results hold across datasets, neural network architectures, and levels of supervision, particularly at low label rates.

2.1 Introduction

Combining graph-based learning with deep learning has achieved state-of-the-art performance in image and remote sensing tasks at low label rates. A common approach is to train unsupervised or semi-supervised neural networks on the raw data, learning a feature space embedding where similarity between features can be used to construct the graph [CCT20, CE22, EHC23, MMS22, CCT23, BOC23, BCH23]. Some works have used graph learning to generate pseudolabels to train a deep neural network in low label rate settings [SAS22, ITA19]. However, neural network training and graph learning are decoupled in these frameworks. A more direct approach, proposed by Wang and Osher, replaced the softmax activation layer of a neural network with graph learning in both training and testing, demonstrating improved generalization and adversarial robustness [WO21]. However, they do not compute the true gradients with respect to the loss, due to difficulties in backpropagating through graph learning, and instead replace them with a heuristic. Our work was partially motivated by these limitations.

A significant gap exists in the literature as no work fully integrates graph learning into the gradient-based learning framework of neural networks. In this chapter, we leverage the adjoint graph Laplace equation to derive the exact backpropagation gradients for similarity graph construction and a general family of graph learning methods, and show how to implement them efficiently. This significantly expands upon previous work [EHC23], where the backpropagation equations were only derived for Laplace learning on a fully connected graph. The backpropagation equations allow us to design a neural network layer that replaces the standard projection head and softmax classifier with a graph learning classifier (see Figure 2.2 for a comparison). We call this novel layer the *Graph Learning Layer* (GLL). A GLL (Figure 2.1) can be combined with *any* neural network architecture and enables end-to-end training in a gradient-based optimization framework. Through a variety of experiments, we show that the GLL learns the intrinsic geometry of data, improves the training dynamics and test

accuracy of both shallow and deep networks across levels of supervision, and is significantly more robust to adversarial attacks compared to a projection head and softmax classifier. Our code is freely available at `https://github.com/jwcalder/GraphLearningLayer`.

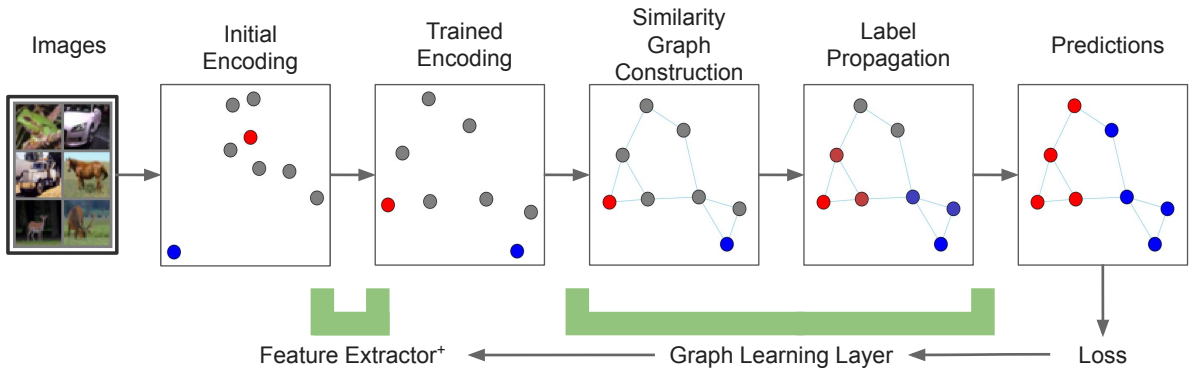
2.1.1 Related Work

We refer the reader to Chapter 1.2.4 for a general overview of related works for GBSSL. Here, we briefly highlight a few additional works relevant to the present Chapter.

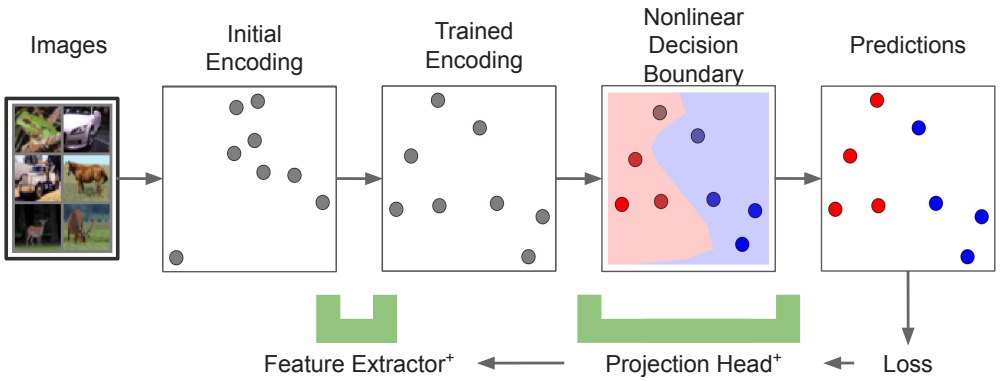
Deep learning methods have increasingly leveraged graph-based approaches or drawn inspiration from them [CWD24]. Some works applied transformers directly to a generated similarity graph for node classification [WZL22, HWG22, ZGG22]. Contrastive learning (CL) uses pair-wise similarity, akin to a complete similarity graph, to facilitate the learning process [CKS20, CKN20, KTW20, WZD23]. Backpropagation techniques have been studied for the differentiation of convex optimization problems [AAB19]. This Chapter extends these ideas to nonlinear equations on graphs involving graph Laplacians.

2.1.2 Our Contributions

The most significant difference between our work and any of the preceding works is the derivation and implementation of the exact backpropagation equations through the Graph Learning Layer, including the similarity graph construction and the graph-based classifier. We also give a sparse, efficient implementation of these equations. This is a significant extension of Enwright et al. [EHC23], where the authors derive the backpropagation gradients for Laplace learning in the special case of a fully connected graph. In this chapter, we derive the gradients for any k-nearest neighbors-type graph construction and any nonlinear elliptic graph learning equation. In addition, we conduct detailed experiments on a variety of benchmark datasets and neural network architectures. Our analysis includes visualizing clustering effects in the latent space under different GLL hyperparameters, demonstrating improved



(a) Graph Learning Layer



(b) Standard MLP projection head

Figure 2.1: The graph learning layer (GLL) within a neural network pipeline compared to a standard multilayer perceptron (MLP) projection head. For the GLL, both labeled and unlabeled input images are batched into the feature extractor, which encodes the images in feature space. The GLL is a combination of two steps; it generates a similarity graph from the encoded data and then propagates the labels across the graph, giving predictions on the unlabeled data. These predictions are used in the desired loss function and gradients can flow through the GLL to update the feature extractor. By contrast, the MLP projection head pipeline does not involve any initially labeled nodes and the classification method ignores relational information in the encodings. The $^+$ indicates learnable parameters in the network; note GLL has none. [BCH24, Figure 1].

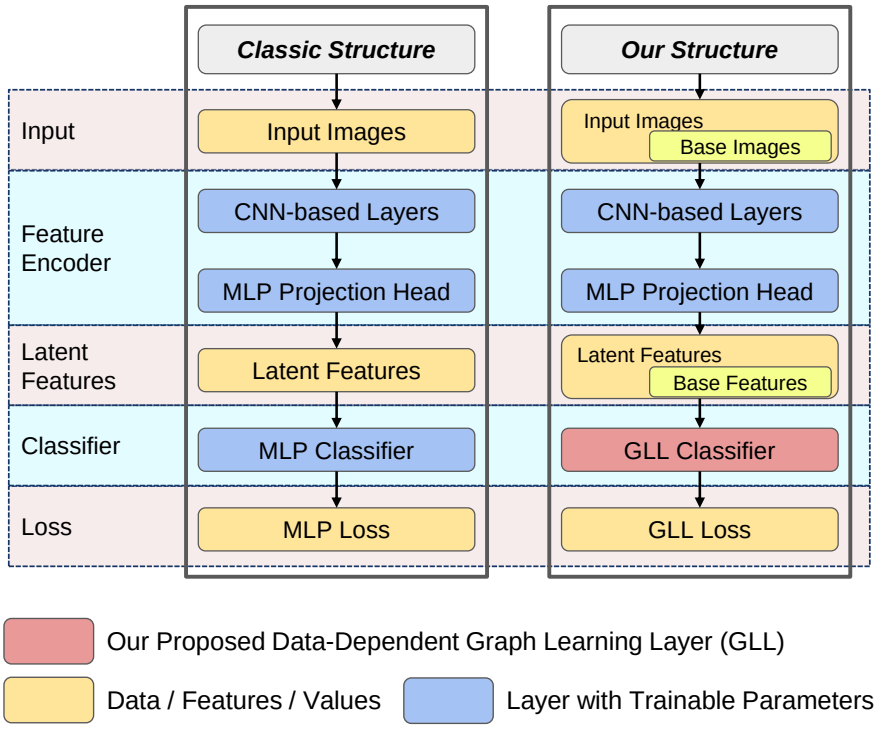


Figure 2.2: Comparison between the classic structure and our proposed graph learning layer (GLL) neural network structures for image classification. Both structures consist of a feature encoder and a classifier, but we replace the multilayer perceptron (MLP) classifier - consisting of one or more linear and activation layers and a final softmax classifier - with our data-dependent GLL with no trainable parameters. We reserve a subsample of the training data - denoted *base* samples - to use as the labeled nodes for label propagation. The samples that are not from the base set are use to compute the loss. [BCH24, Figure 2]

training dynamics and generalization, and showcasing improved robustness to adversarial attacks. In the vast majority experiments - especially at low label rates - a GLL outperforms the standard projection head and softmax-based classifier.

The remainder of the Chapter is organized as follows. Section 2.2 details the necessary background on Laplace Learning, the seminal graph-based learning algorithm. Section 2.3 describes our novel mathematical contributions powering the GLL. Namely, we derive the

gradient of the loss with respect to the similarity graph weight matrix and the associated k-Nearest Neighbor graph construction, which allows us to integrate graph learning into a neural network training context. We also demonstrate the effect this novel GLL has on the learned neural network embeddings. Section 2.4 contains illustrative experiments showcasing the benefits of the GLL, demonstrating improved generalization across network architectures and levels of supervision, and superior robustness to adversarial attacks compared to the softmax classifier.

2.2 Background

In this section, we provide the necessary background on the construction similarity graphs, calculus on graphs, and graph learning using various types of graph Laplace equations. While there is some overlap with Chapter 1.2, for clarity, we restate it here using notation and conventions tailored to the methods developed later in this Chapter.

2.2.1 Similarity Graph Construction

Let $\mathcal{X}$ be the vertex set of a graph with $n \in \mathbb{N}$ vertices and nonnegative edge weights $w_{xy} \geq 0$, for $x, y \in \mathcal{X}$. Let $W = (w_{xy})_{x,y \in \mathcal{X}}$ denote the $n \times n$ weight matrix for the graph. We always assume the weight matrix W is symmetric, so $w_{xy} = w_{yx}$. We usually think of W as a similarity matrix, in which case the entry w_{xy} encodes how similar the nodes x and y are, with large positive values indicating a high degree of similarity, and $w_{xy} = 0$ indicating the nodes are not similar. Given we have a feature vector in $\mathbb{R}^d$ for each node $x \in \mathcal{X}$ in the graph, a common way to construct a similarity weight matrix, which we adopt in the numerical experiments in this chapter, is a k-nearest neighbor graph based on similarity of feature vectors. In this case, we will denote the vertex set of the graph by $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector for the i^{th} node in the graph.

The weight $w_{xy} = w_{\mathbf{x}_i\mathbf{x}_j} = w_{ij}$ between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ is then given by the expression

$$w_{ij} = \exp\left(\frac{-4\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\varepsilon_k(\mathbf{x}_i)\varepsilon_k(\mathbf{x}_j)}\right). \tag{2.2.1}$$

The parameter k is the number of neighbors used in the graph construction, and we choose the bandwidth parameter $\varepsilon_k(\mathbf{x}_i)$ to be the distance from $\mathbf{x}_i$ to its kth nearest neighbor in $\mathbb{R}^d$, which allows the weights to self-tune appropriately to areas of density and sparsity. If desired for applications, $\varepsilon_k(\mathbf{x}_i)$ may be set to a constant value for all $\mathbf{x}_i$. To produce a sparse graph, we set the weight $w_{ij} = 0$ if $\mathbf{x}_i$ is not among the k nearest neighbors of $\mathbf{x}_j$, and $\mathbf{x}_j$ is not among the k nearest neighbors of $\mathbf{x}_i$. While searching for the exact k nearest neighbors in high dimensional data sets is a computational hard problem, there are many fast approximate nearest neighbor algorithms that can be used in practice.[1] While similarity matrices are often used in practice, we note that the theory in this section, and much of Section 2.3, does not place any restrictions on W aside from symmetry $w_{xy} = w_{yx}$.

2.2.2 Calculus on graphs

We let $\ell^2(\mathcal{X})$ denote the Hilbert space of functions $u : \mathcal{X} \to \mathbb{R}^k$, equipped with the inner product

$$\langle u, v\rangle_{\ell^2(\mathcal{X})} = \sum_{x\in\mathcal{X}} u(x)\cdot v(x), \tag{2.2.2}$$

and norm $\|u\|^2_{\ell^2(\mathcal{X})} = \langle u, u\rangle_{\ell^2(\mathcal{X})}$. For $p \geq 1$ we also define p-norm

$$\|u\|^p_{\ell^p(\mathcal{X})} = \sum_{x\in\mathcal{X}} \|u(x)\|^p. \tag{2.2.3}$$

The *degree* is a function $\deg \in \ell^2(\mathcal{X})$ with $k = 1$ defined by

$$\deg(x) = \sum_{y\in\mathcal{X}} w_{xy}.$$

[1] We use the Annoy python package: `https://github.com/spotify/annoy`

We let $\ell^2(\mathcal{X}^2)$ denote the space of functions $\mathbf{u} : \mathcal{X}^2 \to \mathbb{R}^k$. A *vector field* on the graph is a function $\mathbf{u} \in \ell^2(\mathcal{X}^2)$ that is skew symmetric, so $\mathbf{u}(x, y) = -\mathbf{u}(y, x)$. We use bold face for vector fields to distinguish from functions on the graph. The gradient $\nabla u \in \ell^2(\mathcal{X}^2)$ of $u \in \ell^2(\mathcal{X})$ defined by

$$\nabla u(x, y) = u(x) - u(y) \tag{2.2.4}$$

is an example of a vector field over the graph. Note that ∇u is a vector field, so it is defined over the edges in the graph, i.e., pairs (x, y) of vertices, while u is a function of a single vertex x on the graph. For $\mathbf{u}, \mathbf{v} \in \ell^2(\mathcal{X}^2)$ we define an inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\ell^2(\mathcal{X}^2)} = \frac{1}{2} \sum_{x,y \in \mathcal{X}} w_{xy} \mathbf{u}(x, y) \mathbf{v}(x, y), \tag{2.2.5}$$

together with a norm $\|\mathbf{u}\|^2_{\ell^2(\mathcal{X}^2)} = \langle \mathbf{u}, \mathbf{u} \rangle_{\ell^2(\mathcal{X}^2)}$.

The *graph divergence* is an operator taking vector fields to functions in $\ell^2(\mathcal{X})$, and is defined as the adjoint of the gradient. Here, we define the divergence for any function $\mathbf{v} \in \ell^2(\mathcal{X}^2)$; that is, it may not be a vector field. Hence, for $\mathbf{v} \in \ell^2(\mathcal{X}^2)$, the graph divergence $\operatorname{div} \mathbf{v} \in \ell^2(\mathcal{X})$ is defined so that the identity

$$\langle \nabla u, \mathbf{v} \rangle_{\ell^2(\mathcal{X}^2)} = \langle u, \operatorname{div} \mathbf{v} \rangle_{\ell^2(\mathcal{X})} \tag{2.2.6}$$

holds for all $u \in \ell^2(\mathcal{X})$. A straightforward computation shows that

$$\operatorname{div} \mathbf{v}(x) = \frac{1}{2} \sum_{y \in \mathcal{X}} w_{xy} (\mathbf{v}(x, y) - \mathbf{v}(y, x)). \tag{2.2.7}$$

If $\mathbf{v}$ is a vector field, then this can be simplified to

$$\operatorname{div} \mathbf{v}(x) = \sum_{y \in \mathcal{X}} w_{xy} \mathbf{v}(x, y). \tag{2.2.8}$$

The gradient is an operator $\nabla : \ell^2(\mathcal{X}) \to \ell^2(\mathcal{X}^2)$, and the divergence is an operator $\operatorname{div} : \ell^2(\mathcal{X}^2) \to \ell^2(\mathcal{X})$. The graph Laplacian is the composition of these two operators $\Delta = \operatorname{div} \circ \nabla$, which can be expressed as

$$\Delta u(x) = \operatorname{div} \nabla u(x) = \sum_{y \in \mathcal{X}} w_{xy} (u(x) - u(y)). \tag{2.2.9}$$

By the definition $\Delta = \operatorname{div} \circ \nabla$ we have

$$\langle \Delta u, v \rangle_{\ell^2(\mathcal{X})} = \langle \nabla u, \nabla v \rangle_{\ell^2(\mathcal{X}^2)},$$

for any $u, v \in \ell^2(\mathcal{X})$. In particular, we have

$$\langle \Delta u, u \rangle_{\ell^2(\mathcal{X})} = \langle \nabla u, \nabla u \rangle_{\ell^2(\mathcal{X}^2)} = \|\nabla u\|^2_{\ell^2(\mathcal{X}^2)} = \frac{1}{2} \sum_{x,y \in \mathcal{X}} w_{xy}(u(x) - u(y))^2,$$

which is the Dirichlet energy on the graph. For a connected graph with $k = 1$, the kernel of the graph Laplacian Δ is 1-dimensional and spanned by the vector of all ones. In general, the multiplicity of the eigenvalue 0 indicates the number of connected components in the graph [Lux07].

2.2.3 Classical Graph-based Learning

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ be a set of d-dimensional feature vectors, and let W denote a corresponding weight matrix, which may be constructed as a similarity matrix, as discussed in Section 2.2.1. This endows the data set with a graph structure. Let $\mathcal{L} \subset \mathcal{X}$ be the set of labeled indices that identifies which feature vectors have labels $y_i \in \{1, 2, \ldots, k\}$ (for a k-class classification problem). The task of graph-based semi-supervised learning is to utilize the graph structure to infer the labels on the unlabeled index set $\mathcal{U} = \mathcal{X} \setminus \mathcal{L}$. The labels y_i are represented by one-hot vectors; that is, the vector $e_{y_i} \in \mathbb{R}^k$ of all zeros, except a 1 in the y_i^{th} entry. As in Section 2.2.2, generic elements of $\mathcal{X}$, i.e., graph vertices, are denoted as $x, y \in \mathcal{X}$, and the corresponding entry of the weight matrix W is denoted w_{xy}.

Laplace learning [ZGL03], also called *label propagation*, is a seminal method in graph-based learning. The labels are inferred by solving the graph Laplace equation

$$\begin{cases} \Delta u(x) = 0, & \text{if } x \in \mathcal{U} \\ u(x) = g(x), & \text{if } x \in \mathcal{L}, \end{cases} \tag{2.2.10}$$

where $g : \mathcal{L} \to \mathbb{R}^k$ encodes the labeled data through their one hot vector representations. The solution $u \in \ell^2(\mathcal{X})$ is a function $u : \mathcal{X} \to \mathbb{R}^k$ where the components of $u(x) \in \mathbb{R}^k$ can

be interpreted as the probability that x belongs to each class. The label prediction is the class with the highest probability. Equation (2.2.10) is a system of k linear equations, one for each class, which are all completely independent of each other. Thus, this formulation of Laplace learning is equivalent to applying the one-vs-rest method for converting a binary classifier to the multiclass setting.

Laplace learning also has a variational interpretation as the solution to the problem

$$\min_{u \in \ell^2(\mathcal{X})} \frac{1}{2} \sum_{x,y \in \mathcal{X}} w_{xy} \|u(x) - u(y)\|^2 = \|\nabla u\|^2_{\ell^2(\mathcal{X}^2)}, \tag{2.2.11}$$

subject to the condition that $u = g$ on the labeled set $\mathcal{L}$. The necessary condition satisfied by any minimizer of (2.2.11) is exactly the Laplace equation (2.2.10). Furthermore, as long as the graph is connected[2], the solution of (2.2.10) exists and is unique, as is the minimizer of (2.2.11) [Cal18].

There are variants of Laplace learning that employ soft constraints on the agreement with given labels, which can be useful when there is noise or corruption of the labels, typical in many real world settings. Fixing a parameter $\lambda > 0$, the soft-constrained Laplace learning problem minimizes

$$\min_{u \in \ell^2(\mathcal{X})} \left\{ \|\nabla u\|^2_{\ell^2(\mathcal{X}^2)} + \lambda \sum_{x \in \mathcal{L}} \|u(x) - g(x)\|^2 \right\}, \tag{2.2.12}$$

where $\lambda > 0$ tunes the tradeoff between label fidelity and label smoothness. Provided the graph is connected, the unique minimizer of (2.2.12) is the solution of the graph Laplace equation

$$\Delta u + \lambda \xi (u - g) = 0, \tag{2.2.13}$$

where $\xi(x) = 0$ if $x \in \mathcal{U}$ and $\xi(x) = 1$ if $x \in \mathcal{L}$. We recover Laplace learning (2.2.10) in the limit as $\lambda \to \infty$.

[2]In fact, a weaker condition that the graph is connected to the labeled data $\mathcal{L}$ is sufficient, which is equivalent to asking that every connected component of the graph has at least one labeled data point.

The graph Laplace equations (2.2.10) and (2.2.13) are positive definite symmetric linear equations, and can be solved with any number of direct or indirect techniques. When the size of the problem is small, a direct matrix inversion may be used. For larger problems, especially for large sparse systems, an iterative method like the preconditioned conjugate gradient method is preferred.

There are also a variety of graph-based learning techniques that do not require labeled information, such as the Ginzburg-Landau formulation [GMB14]

$$\min_{u\in\ell^2(\mathcal{X})}\left\{\|\nabla u\|^2_{\ell^2(\mathcal{X}^2)}+\lambda\sum_{x\in\mathcal{X}}G(u(x))\right\}, \tag{2.2.14}$$

where G is a many-well potential, with wells located at the one hot label vectors. For binary classification, the wells may be positioned at ± 1 and $G(u) = (1-u^2)^2$. MBO methods are effective approximations of the Ginzburg-Landau energy (2.2.14) when $\lambda \gg 1$ [MKB13, GMB14]. This method was first developed in a semi-supervised setting but also used for modularity optimization [BBT18] and a piecewise constant Mumford-Shah model in an unsupervised setting [HSB15].

2.2.4 Diagonal Perturbation

A modification to Laplace learning (Section 2.2.3) is to add a type of Tikhonov regularization in the energy (see e.g. the work of Miller and Calder [MC23]), leading to the minimization problem

$$\min_{u\in\ell^2(\mathcal{X})}\left\{\|\nabla u\|^2_{\ell^2(\mathcal{X}^2)}+\tau\|u\|^2_{\ell^1(\mathcal{X})}\right\}, \tag{2.2.15}$$

subject to the condition that $u = g$ on the labeled set $\mathcal{L}$. Tikhonov regularization can also be added to the soft constrained Laplace learning (2.2.12). The minimizer of (2.2.15) is the solution of the graph Laplace equation (2.2.10) except that the graph Laplacian, Δ, is replaced by the diagonal perturbed Laplacian $\Delta_\tau = \Delta + \tau I$ (likewise for the soft constrained problem (2.2.13)).

This regularization term increases the convergence speed of iterative algorithm by improving the condition number of the graph Laplacian. Moreover, it induces exponential decay away from labeled data points. This inhibits label propagation and weakens predictions further from labeled data in the latent space, which can limit overconfident predictions. In Section 2.4.2, we show that employing this regularization in the GLL causes denser clustering of similar samples in the latent space learned by the neural network, which may be desireable for classification or clustering tasks.

2.2.5 p-Laplace learning

There are also graph-learning methods based on the nonlinear graph p-Laplacian [ZB11, ECR16, FCL22, ST19, Cal18]. One such example is the variational p-Laplacian, which stems from solving the optimization problem

$$\min_{u \in \ell^2(\mathcal{X})} \frac{1}{2} \sum_{x,y \in \mathcal{X}} w_{xy} \|u(x) - u(y)\|^p, \tag{2.2.16}$$

subject to $u = g$ on $\mathcal{L}$. Taking values of $p \gg 2$ was proposed as a method for improving Laplace learning in the setting of very few labeled data points [ECR16], and theoretical analysis later verified this claim [ST19, Cal18]. The minimizer of the p-variational problem (2.2.16) solves the graph p-Laplace equation

$$\begin{cases} \operatorname{div}(\|\nabla u\|^{p-2} \nabla u)(x) = 0, & \text{if } x \in \mathcal{U} \\ u(x) = g(x), & \text{if } x \in \mathcal{L}. \end{cases} \tag{2.2.17}$$

Another form of the p-Laplacian—the game theoretic p-Laplacian— has also been proposed and studied [Cal18, CD24].

2.2.6 Poisson learning

Another recently proposed approach for low label rate graph-based semi-supervised learning is *Poisson learning* [CCT20]. Instead of solving the graph Laplace equation (2.2.10), Poisson

learning solves the graph Poisson equation

$$\Delta u = \sum_{x \in \mathcal{L}} (g - \overline{g}) \delta_x$$

where $\overline{g} = \frac{1}{m} \sum_{x \in \mathcal{L}} g(x)$, m is the number of labeled examples in $\mathcal{L}$, and $\delta_x(y) = 0$ if $x \neq y$ and $\delta_x(x) = 1$. In Poisson learning, the labels are encoded as point sources and sinks in a graph Poisson equation, which improves the performance at very low label rates [CCT20]. Poisson learning also has the variational interpretation

$$\min_{u \in \ell^2(\mathcal{X})} \left\{ \|\nabla u\|^2_{\ell^2(\mathcal{X}^2)} - \sum_{x \in \mathcal{L}} (g(x) - \overline{g}) \cdot u(x) \right\}, \tag{2.2.18}$$

which is similar to the soft constrained Laplace learning problem (2.2.12), except that here we use a dot product fidelity instead of the $\ell^2(\mathcal{X})$ fidelity.

2.3 Automatic differentiation through graph learning

In this section, we detail our main mathematical contributions, deriving the backpropagation equations necessary for incorporating graph-based learning algorithms into gradient-based learning settings, such as deep learning. Our results are general and apply to any nonlinear elliptic equation on a graph.

We assume the following generic pipeline. In the forward pass, the graph learning layer receives feature representations (i.e. the output of a neural network-based encoder) for a batch of data. Then, a k-nearest neighbors search is conducted over the features and we construct a sparse weight matrix and corresponding graph Laplacian, as described in Section 2.2.1. We then solve a graph Laplace equation, such as Equation (2.2.10), to make label predictions at a set of unlabeled nodes.

In the backward pass, the GLL receives the upstream gradients coming from a loss function and need to return the gradients of the loss with respect to the neural network's feature representations. The calculations for backpropagation can be divided into two distinct steps:

tracking the gradients through the graph Laplace equation with respect to the entries in the weight matrix, which is addressed in Section 2.3.1, and then tracking the gradients from the weight matrix to the feature vectors, which is addressed in Section 2.3.2. In practice, we combine the implementation of these two terms within the same autodifferentiation function, which allows for more efficient calculations.

2.3.1 Backpropagation through graph Laplace equations

In this section, we present the derivation of the backpropagation equations for computing the gradient of the solution of a graph Laplace equation for label propagation with respect to the ingredients of the equation, including the weight matrix used to construct the graph, and any source terms or boundary conditions that are present in the equation. For simplicity, we restrict our attention to binary classification, since the multi-class setting of any graph learning algorithm is equivalent to the one-vs-rest approach.

2.3.1.1 Linear graph Laplace equations

In order to elucidate our approach in a simpler setting, we will first present our main results for the linear graph Laplace equation

$$\begin{cases} \tau u(x) + \Delta u(x) = f(x), & \text{if } x \in \mathcal{U} \\ u(x) = g(x), & \text{if } x \in \mathcal{L}. \end{cases} \tag{2.3.1}$$

Here, $\tau \geq 0$ is a parameter, Δ is the graph Laplacian, defined in (2.2.9), g is the boundary condition, and f is the source term. Depending on the context, the label information can be encoded into the source term f or the boundary condition g.

As discussed previously, we have in mind that a graph PDE like (2.3.1) is a layer within a neural network that is trained end to end. The inputs to the graph learning layer are any one of the following objects (or all of them):

- The weight matrix entries w_{xy}.

- The source term f.
- The boundary condition g.

In all cases, the output of the graph learning layer is the solution $u \in \ell^2(\mathcal{X})$, which feeds into the next layer of the network. Let us keep in mind that, while we have used functional analysis notation to illuminate the main ideas more clearly, the inputs and outputs of the block are all vectors in Euclidean space; i.e., a function $u \in \ell^2(\mathcal{X})$ is simply a vector in $\mathbb{R}^n$ where n is the number of nodes in the graph.

The graph learning layer is part of an end-to-end network that is fed into a scalar loss function that we denote by $\mathcal{J}$. For $u \in \ell^2(\mathcal{X})$ the solution of (2.3.1), let $\nabla_u \mathcal{J} \in \ell^2(\mathcal{X})$ be the gradient of the loss $\mathcal{J}$ with respect to the output u of the graph learning layer, so

$$\nabla_u \mathcal{J}(x) = \partial_{u(x)} \mathcal{J},$$

where ∂_z denotes a scalar partial derivative in the scalar variable z. Backpropagating through a graph learning layer requires taking as input the gradient $\nabla_u \mathcal{J}$ and computing, via the chain rule, the gradients

$$\nabla_W \mathcal{J}(x,y) = \left\langle \partial_{w_{xy}} u, \nabla_u \mathcal{J} \right\rangle_{\ell^2(\mathcal{X})} \tag{2.3.2}$$

$$\nabla_f \mathcal{J}(x) = \left\langle \partial_{f(x)} u, \nabla_u \mathcal{J} \right\rangle_{\ell^2(\mathcal{X})}, \quad \text{and} \tag{2.3.3}$$

$$\nabla_g \mathcal{J}(x) = \left\langle \partial_{g(x)} u, \nabla_u \mathcal{J} \right\rangle_{\ell^2(\mathcal{X})}, \tag{2.3.4}$$

for all $x, y \in \mathcal{X}$. The quantities $\nabla_W \mathcal{J}$, $\nabla_f \mathcal{J}$, and $\nabla_g \mathcal{J}$ are the outputs of the graph learning autodifferentiation procedure and are then fed into the preceding autodifferentiation block. Letting m be the number of labeled points, $\nabla_W \mathcal{J}$ is an $n \times n$ matrix, while $\nabla_f \mathcal{J}$ is a length $n - m$ vector and $\nabla_g \mathcal{J}$ is a length m vector.

In the special case of the linear graph Laplace equation (2.3.1), backpropgation through the graph learning layer requires solving the *adjoint* equation

$$\begin{cases} \tau v(x) + \Delta v(x) = \nabla_u \mathcal{J}, & \text{if } x \in \mathcal{U} \\ v(x) = 0, & \text{if } x \in \mathcal{L} \end{cases} \tag{2.3.5}$$

for the unknown function $v \in \ell^2(\mathcal{X})$. Since the left hand side of (2.3.1) is self-adjoint, the adjoint equation (2.3.5) is nearly identical to the original equation (2.3.1) — indeed, the main difference is that the boundary conditions have been set to zero, and the source term is replaced by the upstream gradient $\nabla_u \mathcal{J}$, which is the input to the automatic differentiation method. Hence, the computational complexity of the adjoint equation is similar to the original graph Laplace equation that was solved in the forward pass.

In this case, the adjoint equation (2.3.5) is uniquely solvable for v, provided the graph is connected, and the downstream gradients we wish to compute can all be expressed in terms of v as follows (see Theorem 2.3.5):

$$\nabla_W \mathcal{J}(x,y) = -\nabla u(x,y) v(x) = -(u(x) - u(y))v(x) \quad \text{for } x, y \in \mathcal{X}, \tag{2.3.6}$$

$$\nabla_f \mathcal{J}(x) = v(x) \quad \text{for } x \in \mathcal{X} \setminus \mathcal{L} \tag{2.3.7}$$

$$\nabla_g \mathcal{J}(x) = \nabla_u \mathcal{J}(x) - \Delta v(x) \quad \text{for } x \in \mathcal{L}. \tag{2.3.8}$$

We note that the last expression for $\nabla_g \mathcal{J}(x)$ cannot be simplified using the adjoint equation, since it is evaluated on the labeled nodes $\mathcal{L}$ where the adjoint equation does not hold. We also note that it may seem a little strange at first that $\nabla_W \mathcal{J}(x,y)$ is not symmetric in x and y. However, as we show in the next section, since the weight matrix W is symmetric, we can replace $\nabla_W \mathcal{J}(x,y)$ by the symmetrized expression

$$-\nabla u(x,y) \nabla v(x,y) = -(u(x) - u(y))(v(x) - v(y))$$

without affecting any downstream gradient computations. We use this symmetrized expression in practice to simplify computations.

The proofs of the claims in this section are all provided in a more general setting in Theorem 2.3.5 in the following section.

2.3.1.2 Nonlinear elliptic equations on graphs

In order to make our results general - applicalbe to any of the graph-based learning algorithms discussed in Section 2.2, including nonlinear graph-Laplace equations - we proceed in this section with generalizing the simple graph Laplace equation (2.3.1) to a general nonlinear elliptic equation on a graph. Given a function $\varphi : \mathbb{R} \times \mathcal{X}^2 \to \mathbb{R}$ and a vector field $\mathbf{v} \in \ell^2(\mathcal{X}^2)$, we define the function $\varphi(\mathbf{v}, \cdot) \in \ell^2(\mathcal{X}^2)$ by

$$\varphi(\mathbf{v}, \cdot)(x, y) = \varphi(\mathbf{v}(x, y), x, y).$$

Let us write $\varphi = \varphi(q, x, y)$ and write φ_q for the partial derivative of φ in q, when it exists. In general, $\varphi(\mathbf{v}, \cdot)$ may not be a vector field, without further assumptions on φ.

Definition 2.3.1. *We say* $\varphi : \mathbb{R} \times \mathcal{X}^2 \to \mathbb{R}$ preserves vector fields *if*

$$\varphi(-q, x, y) = -\varphi(q, y, x)$$

holds for all $q \in \mathbb{R}$ *and* $x, y \in \mathcal{X}$.

Clearly, if φ preserves vector fields, then for any vector field $\mathbf{v}$, $\varphi(\mathbf{v}, \cdot)$ is also a vector field.

We also define ellipticity and symmetry.

Definition 2.3.2. *We say that* $\varphi : \mathbb{R} \times \mathcal{X}^2 \to \mathbb{R}$ *is* elliptic *if* $q \mapsto \varphi(q, x, y)$ *is monotonically nondecreasing for all* $x, y \in \mathcal{X}$.

Definition 2.3.3. *We say that* $\varphi : \mathbb{R} \times \mathcal{X}^2 \to \mathbb{R}$ *is* symmetric *if*

$$\varphi_q(q, x, y) = \varphi_q(-q, y, x)$$

holds for all $q \in \mathbb{R}$ *and* $x, y \in \mathcal{X}$ *for which the derivatives* $\varphi_q(q, x, y)$ *and* $\varphi_q(-q, y, x)$ *exist.*

The notion of ellipticity is also called monotonicity in some references [BS91, CE22], and has appeared previously in the study of graph partial differential equations [MOS15, Cal18].

We will consider here a nonlinear graph Laplace-type equation of the form

$$\begin{cases} \sigma(u(x), x) + \operatorname{div} \varphi(\nabla u, \cdot)(x) = f(x), & \text{if } x \in \mathcal{X} \setminus \mathcal{L} \\ u(x) = g(x), & \text{if } x \in \mathcal{L}, \end{cases} \tag{2.3.9}$$

where $\sigma : \mathbb{R} \times \mathcal{X} \to \mathbb{R}$, $f \in \ell^2(\mathcal{X})$, $\mathcal{L} \subset \mathcal{X}$, and $g : \mathcal{X} \to \mathbb{R}$. The graph Laplace equation generalizes all of the examples of graph Laplace-based semi-supervised learning algorithms given in Section 2.2. In particular, we allow for the situation that $\mathcal{L} = \varnothing$. We will assume the equation (2.3.9) is uniquely solvable; see the work of Calder and Ettehad [CE22] for conditions under which this holds. However, at the moment we do not place any assumptions on σ and φ; in particular, φ need not be elliptic, preserve vector fields, or be symmetric.

Example 1. *An example is the graph p-Laplace equation for $p \geq 1$, where $\sigma(z, x) = \tau z$ for a nonnegative constant τ, $f \in \ell^2(\mathcal{X})$, and*

$$\varphi(q, x, y) = |q|^{p-2} q.$$

In this case, the equation in (2.3.9) *becomes*

$$\tau u(x) + \operatorname{div}(|\nabla u|^{p-2} \nabla u)(x) = f(x)$$

or written out more explicitly

$$\tau u(x) + \sum_{y \in \mathcal{X}} w_{xy} |u(x) - u(y)|^{p-2} (u(x) - u(y)) = f(x).$$

Since $\varphi_q(q, x, y) = (p-1)|q|^{p-2}$, we have that φ preserves vector fields, and is both elliptic and symmetric for $p \geq 1$. Note that $p = 2$ recovers Laplace learning*, which will be the focus of our experimental Section 2.4.*

Remark 2.3.4. If φ is elliptic, then the divergence part of (2.3.9) arises from the minimization of a convex energy on the graph of the form

$$\sum_{x,y \in \mathcal{X}} w_{xy} \psi(u(x) - u(y), x, y),$$

where ψ is any antiderivative of φ in the q variable. The ellipticity guarantees that ψ_q is nondecreasing, and so ψ is convex in q. For example, with the usual graph Laplacian, $\varphi(q) = \frac{1}{2}q$ and $\psi(q) = q^2$.

As we show below, efficient computation of the gradients (2.3.2), (2.3.3), and (2.3.4)—the essence of backpropagation—involves solving the *adjoint* equation

$$\begin{cases} \sigma_z(u(x), x)v(x) + \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right) = \nabla_u \mathcal{J}(x), & \text{if } x \in \mathcal{X} \setminus \mathcal{L} \\ v(x) = 0, & \text{if } x \in \mathcal{L} \end{cases} \tag{2.3.10}$$

for the unknown function $v \in \ell^2(\mathcal{X})$. Recall that σ_z denotes the partial derivative of the function $(z, x) \mapsto \sigma(z, x)$ in the first variable z, i.e., if $\sigma(z, x) = z^2$ then $\sigma_z(z, x) = 2z$. We also note that using the definition of divergence given in (2.2.8), the divergence term $\operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right)$ evaluated at a vertex $x \in \mathcal{X} \setminus \mathcal{L}$ in (2.3.10) can be expressed as

$$\frac{1}{2}\sum_{y \in \mathcal{X}} w_{xy}(\varphi_q(\nabla u(x, y), x, y)(v(x) - v(y)) - \varphi_q(\nabla u(y, x), y, x)(v(y) - v(x))). \tag{2.3.11}$$

If φ is symmetric, as per Definition 2.3.3, then $\varphi_q(-q, y, x) = \varphi_q(q, x, y)$ and this expression can be simplified to read

$$\operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right) = \sum_{y \in \mathcal{X}} w_{xy}\, \varphi_q(\nabla u(x, y), x, y)(v(x) - v(y)). \tag{2.3.12}$$

This expression is a graph Laplacian applied to the function v, with weights that depend on the gradient of u. In fact, the first term $\sigma_z(u(x), x)v(x)$ in the adjoint equation (2.3.10) also depends on the solution u of (2.3.9), which was computed in the forward pass of the neural network and can be saved for use in backpropagation. We also note that, even though the graph PDE (2.3.9) is nonlinear, the adjoint equation (2.3.10) is always a linear equation for v.

The following theorem is our main result for backpropagation through graph learning layers.

Theorem 2.3.5. *Assume $z \mapsto \sigma(z,x)$ and $q \mapsto \varphi(q,x,y)$ are continuously differentiable for all $x, y \in \mathcal{X}$. Let $f \in \ell^2(\mathcal{X})$, $g : \mathcal{L} \to \mathbb{R}$, and suppose $u \in \ell^2(\mathcal{X})$ satisfies (2.3.9). If the adjoint equation (2.3.10) admits a unique solution $v \in \ell^2(\mathcal{X})$, then the partial derivatives $\partial_{w_{xy}} u, \partial_{f(x)} u, \partial_{g(x)} u \in \ell^2(\mathcal{X})$ exist and are continuous in the variables $w_{xy}, f(x)$ and $g(x)$, respectively. Furthermore, it holds that*

$$\nabla_W \mathcal{J}(x,y) = -\frac{1}{2}\Big(\varphi(\nabla u(x,y),x,y) - \varphi(\nabla u(y,x),y,x)\Big)v(x) \quad \text{for } x,y \in \mathcal{X}, \tag{2.3.13}$$

$$\nabla_f \mathcal{J}(x) = v(x) \quad \text{for } x \in \mathcal{X} \setminus \mathcal{L} \tag{2.3.14}$$

$$\nabla_g \mathcal{J}(x) = \nabla_u \mathcal{J}(x) - \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right)(x) \quad \text{for } x \in \mathcal{L}. \tag{2.3.15}$$

Proof. The existence and continuity of the partial derivatives $\partial_{w_{xy}} u, \partial_{f(x)} u, \partial_{g(x)} u \in \ell^2(\mathcal{X})$ follows from the unique solvability of the adjoint equation (2.3.10) and the implicit function theorem.

Now, the adjoint equation is utilized throughout the whole proof in a similar way. Given $h \in \ell^2(\mathcal{X})$ with $h(x) = 0$ for $x \in \mathcal{L}$, we can compute the inner product with $\nabla_u \mathcal{J}$ as follows:

$$\langle h, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} = \langle \sigma_z(u,\cdot)h + \operatorname{div}\left(\varphi_q(\nabla u,\cdot)\nabla h\right), v \rangle_{\ell^2(\mathcal{X})}. \tag{2.3.16}$$

To see this we compute

$$\begin{aligned}
\langle h, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} &= \langle h, \sigma_z(u,\cdot)v + \operatorname{div}\left(\varphi_q(\nabla u,\cdot)\nabla v\right)\rangle_{\ell^2(\mathcal{X})} \\
&= \langle h, \sigma_z(u,\cdot)v\rangle_{\ell^2(\mathcal{X})} + \langle h, \operatorname{div}\left(\varphi_q(\nabla u,\cdot)\nabla v\right)\rangle_{\ell^2(\mathcal{X})} \\
&= \langle \sigma_z(u,\cdot)h, v\rangle_{\ell^2(\mathcal{X})} + \langle \nabla h, \varphi_q(\nabla u,\cdot)\nabla v\rangle_{\ell^2(\mathcal{X}^2)} \\
&= \langle \sigma_z(u,\cdot)h, v\rangle_{\ell^2(\mathcal{X})} + \langle \varphi_q(\nabla u,\cdot)\nabla h, \nabla v\rangle_{\ell^2(\mathcal{X}^2)} \\
&= \langle \sigma_z(u,\cdot)h, v\rangle_{\ell^2(\mathcal{X})} + \langle \operatorname{div}\left(\varphi_q(\nabla u,\cdot)\nabla h\right), v\rangle_{\ell^2(\mathcal{X})} \\
&= \langle \sigma_z(u,\cdot)h + \operatorname{div}\left(\varphi_q(\nabla u,\cdot)\nabla h\right), v\rangle_{\ell^2(\mathcal{X})}.
\end{aligned}$$

We will use the identity (2.3.16) multiple times in what follows.

We now prove (2.3.13). Given $r, s \in \mathcal{X}$, we let $h(x) = \partial_{w_{rs}} u(x)$. Then differentiating both sides of (2.3.9) we find that

$$\sigma_z(u(x), x)h(x) + \operatorname{div}(\varphi_q(\nabla u, \cdot)\nabla h) = -\frac{1}{2}\delta_r(x)(\varphi(\nabla u(r,s), r, s) - \varphi(\nabla u(s,r), s, r)) \quad (2.3.17)$$

for $x \in \mathcal{X} \setminus \mathcal{L}$, and $v(x) = 0$ for $x \in \mathcal{L}$, where $\delta_r(x) = 0$ if $r \neq x$, and $\delta_r(r) = 1$. By (2.3.2), (2.3.16), and (2.3.17) we have

$$\begin{aligned}
\nabla_W \mathcal{J}(r,s) &= \langle \partial_{w_{rs}} u, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} = \langle h, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} \\
&= \langle \sigma_z(u, \cdot)h + \operatorname{div}(\varphi_q(\nabla u, \cdot)\nabla h), v \rangle_{\ell^2(\mathcal{X})} \\
&= -\frac{1}{2} \langle \delta_r(\varphi(\nabla u(r,s), r, s) - \varphi(\nabla u(s,r), s, r)), v \rangle_{\ell^2(\mathcal{X})} \\
&= -\frac{1}{2}(\varphi(\nabla u(r,s), r, s) - \varphi(\nabla u(s,r), s, r))v(r).
\end{aligned}$$

To prove (2.3.14), let $r \in \mathcal{X} \setminus \mathcal{L}$ and set $h(x) = \partial_{f(r)} u(x)$. Then differentiating both sides of (2.3.9) we find that

$$\sigma_z(u(x), x)h(x) + \operatorname{div}(\varphi_q(\nabla u, \cdot)\nabla h) = \delta_r(x), \quad (2.3.18)$$

for $x \in \mathcal{X} \setminus \mathcal{L}$, and $h(x) = 0$ for $x \in \mathcal{L}$. By (2.3.3), (2.3.16), and (2.3.18) we have

$$\nabla_f \mathcal{J}(r) = \langle h, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} = \langle \sigma_z(u, \cdot)h + \operatorname{div}(\varphi_q(\nabla u, \cdot)\nabla h), v \rangle_{\ell^2(\mathcal{X})} = \langle \delta_r, v \rangle_{\ell^2(\mathcal{X})} = v(r).$$

Finally, to prove (2.3.15), let $r \in \mathcal{L}$ and set $h(x) = \partial_{g(r)} u(x)$. Then differentiating both sides of (2.3.9) we find that

$$\sigma_z(u(x), x)h(x) + \operatorname{div}(\varphi_q(\nabla u, \cdot)\nabla h) = 0, \quad (2.3.19)$$

for $x \in \mathcal{X} \setminus \mathcal{L}$ and $h(x) = \delta_r(x)$ for $x \in \mathcal{L}$. By (2.3.4), (2.3.16), and (2.3.19) we have

$$\begin{aligned}
\nabla_g \mathcal{J}(r) &= \langle h, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} = \langle h - \delta_r, \nabla_u \mathcal{J} \rangle_{\ell^2(\mathcal{X})} + \nabla_u \mathcal{J}(r) \\
&= \langle h - \delta_r, \sigma_z(u, \cdot)v + \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right) \rangle_{\ell^2(\mathcal{X})} + \nabla_u \mathcal{J}(r) \\
&= \langle h, \sigma_z(u, \cdot)v + \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right) \rangle_{\ell^2(\mathcal{X})} - \sigma_z(u(r), r)v(r) \\
&\qquad\qquad - \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right)(r) + \nabla_u \mathcal{J}(r) \\
&= \langle \sigma_z(u, \cdot)h + \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla h\right), v \rangle_{\ell^2(\mathcal{X})} - \sigma_z(u(r), r)v(r) \\
&\qquad\qquad - \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right)(r) + \nabla_u \mathcal{J}(r) \\
&= -\sigma_z(u(r), r)v(r) - \operatorname{div}\left(\varphi_q(\nabla u, \cdot)\nabla v\right)(r) + \nabla_u \mathcal{J}(r).
\end{aligned}$$

We finally use that $v(r) = 0$ for $r \in \mathcal{L}$. □

Recall that we have assumed the weight matrix W is symmetric, so $w_{xy} = w_{yx}$. Thus, it may at first seem strange that the gradient $\nabla_W \mathcal{J}(x, y)$ is not symmetric in x and y, but this is a natural consequence of the lack of any restrictions on the variations in W when computing gradients. However, it turns out that we can symmetrize the expression for the gradient $\nabla_W \mathcal{J}(x, y)$ without affecting the backpropagation of gradients, provided that W is guaranteed to be symmetric under perturbations in any variables that it depends upon. This is encapsulated in the following lemma.

Lemma 2.3.6. *Suppose that the weight matrix $W = (w_{xy})_{x,y \in \mathcal{X}}$ depends on a parameter vector $\theta \in \mathbb{R}^m$, so $W = W(\theta)$ and $w_{xy} = w_{xy}(\theta)$ for all $x, y \in \mathcal{X}$, and furthermore that the loss $\mathcal{J}$ depends on θ only through the weight matrix W. If $w_{xy}(\theta) = w_{yx}(\theta)$ for all $\theta \in \mathbb{R}^m$ and $x, y \in \mathcal{X}$ then it holds that*

$$\nabla_\theta \mathcal{J} = \sum_{x,y \in \mathcal{X}} \nabla_W \mathcal{J}(x, y) \nabla_\theta w_{xy} = \sum_{x,y \in \mathcal{X}} \frac{1}{2} \left(\nabla_W \mathcal{J}(x, y) + \nabla_W \mathcal{J}(y, x) \right) \nabla_\theta w_{xy}. \tag{2.3.20}$$

Proof. By the chain rule we have

$$\nabla_\theta \mathcal{J} = \sum_{x,y \in \mathcal{X}} \nabla_W \mathcal{J}(x, y) \nabla_\theta w_{xy} = \sum_{x,y \in \mathcal{X}} \frac{1}{2} \left(\nabla_W \mathcal{J}(x, y) \nabla_\theta w_{xy} + \nabla_W \mathcal{J}(y, x) \nabla_\theta w_{yx} \right).$$

Since $w_{xy}(\theta) = w_{yx}(\theta)$ we have $\nabla_\theta w_{xy} = \nabla_\theta w_{yx}$, which upon substituting above completes the proof. □

Lemma 2.3.6 allows us to replace the gradient $\nabla_W \Psi$ with a symmetrized version without affecting the correct computation of gradients.

Remark 2.3.7. By Lemma 2.3.6, we may replace $\nabla_W \mathcal{J}(x, y)$, given by (2.3.13) with its symmetrized version

$$\frac{1}{2}\left(\nabla_W \mathcal{J}(x, y) + \nabla_W \mathcal{J}(y, x)\right) = -\frac{1}{2}(\varphi(\nabla u(x, y), x, y) - \varphi(\nabla u(y, x), y, x))\nabla v(x, y), \quad (2.3.21)$$

without affecting the resultant backpropagation. The resulting expression is symmetric, in x and y, which may be convenient in implementations.

Example 2. *As in Example 1, we consider the graph p-Laplace equation with* $p > 1$,[3] $\sigma(z, x) = \tau z$, *where* τ *a nonnegative constant, and* $f \in \ell^2(\mathcal{X})$. *In this case we have* $\varphi(q, x, y) = |q|^{p-2}q$ *and* $\varphi_q(q, x, y) = (p - 1)|q|^{p-2}$. *The adjoint equation* (2.3.10) *becomes*

$$\tau v + (p - 1)\operatorname{div}(|\nabla u|^{p-2}\nabla v) = \nabla_u \mathcal{J},$$

subject to $v = 0$ *on* $\mathcal{L}$. *The adjoint equation is uniquely solvable if* $\tau > 0$*; otherwise the solvability depends crucially on the structure of* $|\nabla u|^{p-2}$ *when* $p \neq 2$. *For example, if* u *is constant, so* $\nabla u = 0$, *then the adjoint equation is* not *solvable when* $\tau = 0$. *When* $p = 2$ *and* $\mathcal{L} \neq \varnothing$ *(this is the setting of* Laplace learning*) the adjoint equation is always uniquely solvable, and is simply given by the Laplace equation*

$$\tau v + \Delta v = \nabla_u \Psi,$$

subject to $v = 0$ *on* $\mathcal{L}$.

[3]The case $p = 1$ does not satisfy the continuous differentiability conditions of Theorem 2.3.5.

Proceeding under the assumption that the adjoint equation is uniquely solvable, the expressions for the gradients in Theorem 2.3.5, taking the symmetrized version for $\nabla_W \mathcal{J}$, become $\nabla_f \mathcal{J}(x) = v(x)$ for $x \in \mathcal{X} \setminus \mathcal{L}$,

$$\frac{1}{2}\left(\nabla_W \mathcal{J}(x,y) + \nabla_W \mathcal{J}(y,x)\right) = -|\nabla u(x,y)|^{p-2}\nabla u(x,y)\nabla v(x,y), \tag{2.3.22}$$

for $x, y \in \mathcal{X}$, and

$$\nabla_g \mathcal{J}(x) = \nabla_u \mathcal{J}(x) - (p-1)\,\mathrm{div}\left(|\nabla u|^{p-2}\nabla v\right)(x) \quad \text{for } x \in \mathcal{L}. \tag{2.3.23}$$

When $p = 2$, the symmetrized gradient $\nabla_W \Psi$ for Laplace learning has the especially simple form $-\nabla u(x,y)\nabla v(x,y)$.

2.3.2 Backpropagation through similarity matrices

We now consider the backpropagation of gradients through the construction of the graph weight matrix.

As in Section 2.2.1, we consider a symmetric k-nearest neighbor type graph construction for the weight matrix W, of the form

$$w_{ij} = a_{ij}\eta\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\varepsilon_i\varepsilon_j}\right), \tag{2.3.24}$$

where η is differentiable and a_{ij} is the adjacency matrix of the graph, so $a_{ij} = 1$ if there is an edge between nodes i and j, and $a_{ij} = 0$ otherwise. We consider graphs without self-loops, so $w_{ii} = a_{ii} = 0$ for all i.

In deriving the backpropagation equations, we will assume that the adjacency matrix a_{ij} does not depend on the features $\mathbf{x}_i$. This is an approximation that is true generically for k-nearest neighbor graphs, unless there is a tie for the kth neighbor of some $\mathbf{x}_i$ that allows the neighbor information to change under small perturbation. However, the self-tuning parameters ε_i and ε_j do depend on the feature vectors; in particular, they are defined by

$$\varepsilon_i = \|\mathbf{x}_i - \mathbf{x}_{k_i}\|, \tag{2.3.25}$$

where k_i is the index of the k^{th} nearest neighbor of the vertex i. However, we note that the derivation below does not place any assumptions on the indices $k_1, \dots, k_n$ except that $k_i \neq i$ and $a_{ik_i} = 1$. This, in particular, allows our analysis to handle approximate nearest neighbor searches, where some neighbors may be incorrect, as long as we use the correct distances. We will also assume that the indices k_i do not depend on (or rather, are constant with respect to) any of the features $\mathbf{x}_j$, which is again true under small perturbations of the features, as long as there is no tie for the k^{th} nearest neighbor of any given point. This is generically true for e.g. random data on the unit sphere in $\mathbb{R}^d$.

In backpropagation, we are given the gradients $g_{ij} = \partial_{w_{ij}} \mathcal{J}$ computed in Section 2.3.1, which we can assemble into a matrix $G = (g_{ij})_{ij}$, and we need to compute $\nabla_{\mathbf{x}_\ell} \mathcal{J}$ in terms of the matrix G, where $\mathbf{x}_\ell$ is an arbitrary feature vector. That is to say, we backpropagate the gradients through the graph construction (2.3.24). Our main result is the following lemma.

Lemma 2.3.8. *We have*

$$\nabla_{\mathbf{x}_i} \mathcal{J} = \sum_{j=1}^{n} \frac{\widetilde{g}_{ij} a_{ij} p_{ij}}{\varepsilon_i \varepsilon_j} (\mathbf{x}_i - \mathbf{x}_j) + \sum_{j=1}^{n} b_j (\mathbf{x}_j - \mathbf{x}_{k_j}) \delta_{ik_j} - b_i (\mathbf{x}_i - \mathbf{x}_{k_i}), \tag{2.3.26}$$

where $\widetilde{g}_{ij} = g_{ij} + g_{ji}$, *and the vector* $b = (b_i)_i$ *and matrix* $P = (p_{ij})_{ij}$ *are defined by*

$$b_i = \sum_{j=1}^{n} \frac{\widetilde{g}_{ij} a_{ij} p_{ij}}{2\varepsilon_i^3 \varepsilon_j} \|\mathbf{x}_i - \mathbf{x}_j\|^2, \quad \textit{and} \quad p_{ij} = \eta' \left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\varepsilon_i \varepsilon_j} \right), \tag{2.3.27}$$

where η' *is the derivative of* η.

Remark 2.3.9. Notice in (2.3.26) and (2.3.27), the gradients $\widetilde{g}_{ij}$ always appear next to the corresponding adjacency matrix entry a_{ij}. Thus, in an end to end graph learning block, where the features vectors $\mathbf{x}_i$ are taken as input, we only need to compute the gradients g_{ij} along *edges* in the graph. This allows us to preserve the sparsity structure of the graph during backpropagation, drastically accelerating computations.

Proof. We begin by using the chain rule to compute

$$\nabla_{\mathbf{x}_\ell} \mathcal{J} = \sum_{i,j=1}^{n} \frac{\partial \mathcal{J}}{\partial w_{ij}} \nabla_{\mathbf{x}_\ell} w_{ij} = \sum_{i,j=1}^{n} g_{ij} \nabla_{\mathbf{x}_\ell} w_{ij} = \sum_{i,j=1}^{n} g_{ij} a_{ij} \nabla_{\mathbf{x}_\ell} \eta \left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\varepsilon_i \varepsilon_j} \right). \tag{2.3.28}$$

Notice that the sum above is only over edges in the graph where $a_{ij} = 1$, and we may also omit $i = j$, since the gradient term is zero there. Thus, we need to compute

$$b_{ij}^{\ell} := \nabla_{\mathbf{x}_\ell} \eta \left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\varepsilon_i \varepsilon_j} \right)$$

for $i \neq j$ with $a_{ij} = 1$. Recalling (2.3.27) we have

$$b_{ij}^{\ell} = \frac{p_{ij}}{\varepsilon_i \varepsilon_j} (\mathbf{x}_i - \mathbf{x}_j)(\delta_{i\ell} - \delta_{j\ell}) - \frac{p_{ij}}{2\varepsilon_i^2 \varepsilon_j^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \nabla_{\mathbf{x}_\ell} (\varepsilon_i \varepsilon_j).$$

We now note that

$$\nabla_{\mathbf{x}_\ell} \varepsilon_i = \frac{1}{\varepsilon_i} (\mathbf{x}_i - \mathbf{x}_{k_i})(\delta_{\ell i} - \delta_{\ell k_i}) \quad \text{and} \quad \nabla_{\mathbf{x}_\ell} \varepsilon_j = \frac{1}{\varepsilon_j} (\mathbf{x}_j - \mathbf{x}_{k_j})(\delta_{\ell j} - \delta_{\ell k_j}),$$

so

$$\nabla_{\mathbf{x}_\ell} (\varepsilon_i \varepsilon_j) = \frac{\varepsilon_i}{\varepsilon_j} (\mathbf{x}_j - \mathbf{x}_{k_j})(\delta_{\ell j} - \delta_{\ell k_j}) + \frac{\varepsilon_j}{\varepsilon_i} (\mathbf{x}_i - \mathbf{x}_{k_i})(\delta_{\ell i} - \delta_{\ell k_i}).$$

Plugging this in above we have

$$\begin{aligned} b_{ij}^{\ell} = \frac{p_{ij}}{\varepsilon_i \varepsilon_j} (\mathbf{x}_i - \mathbf{x}_j)(\delta_{i\ell} - \delta_{j\ell}) &- \frac{p_{ij}}{2\varepsilon_i \varepsilon_j^3} \|\mathbf{x}_i - \mathbf{x}_j\|^2 (\mathbf{x}_j - \mathbf{x}_{k_j})(\delta_{\ell j} - \delta_{\ell k_j}) \\ &- \frac{p_{ij}}{2\varepsilon_i^3 \varepsilon_j} \|\mathbf{x}_i - \mathbf{x}_j\|^2 (\mathbf{x}_i - \mathbf{x}_{k_i})(\delta_{\ell i} - \delta_{\ell k_i}). \end{aligned}$$

Plugging this into (2.3.28) we have

$$\begin{aligned} \nabla_{\mathbf{x}_\ell} \mathcal{J} &= \sum_{i,j=1}^{n} g_{ij} a_{ij} b_{ij}^{\ell} \\ &= \sum_{i,j=1}^{n} g_{ij} a_{ij} \frac{p_{ij}}{\varepsilon_i \varepsilon_j} (\mathbf{x}_i - \mathbf{x}_j)(\delta_{i\ell} - \delta_{j\ell}) \\ &\quad - \sum_{i,j=1}^{n} g_{ij} a_{ij} \frac{p_{ij}}{2\varepsilon_i \varepsilon_j^3} \|\mathbf{x}_i - \mathbf{x}_j\|^2 (\mathbf{x}_j - \mathbf{x}_{k_j})(\delta_{\ell j} - \delta_{\ell k_j}) \\ &\quad - \sum_{i,j=1}^{n} g_{ij} a_{ij} \frac{p_{ij}}{2\varepsilon_i^3 \varepsilon_j} \|\mathbf{x}_i - \mathbf{x}_j\|^2 (\mathbf{x}_i - \mathbf{x}_{k_i})(\delta_{\ell i} - \delta_{\ell k_i}) \\ &= \sum_{i=1}^{n} \frac{\widetilde{g}_{\ell i} a_{\ell i} p_{\ell i}}{\varepsilon_\ell \varepsilon_i} (\mathbf{x}_\ell - \mathbf{x}_i) - \left[\sum_{i=1}^{n} \frac{\widetilde{g}_{\ell i} a_{\ell i} p_{\ell i}}{2\varepsilon_\ell^3 \varepsilon_i} \|\mathbf{x}_\ell - \mathbf{x}_i\|^2 \right] (\mathbf{x}_\ell - \mathbf{x}_{k_\ell}) \\ &\qquad - \sum_{i=1}^{n} \left[\sum_{j=1}^{n} \frac{\widetilde{g}_{ij} a_{ij} p_{ij}}{2\varepsilon_i^3 \varepsilon_j} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \right] (\mathbf{x}_{k_i} - \mathbf{x}_i) \delta_{\ell k_i}, \end{aligned}$$

where $\widetilde{g}_{ij} = g_{ij} + g_{ji}$. Recalling (2.3.27), the gradient above can be expressed as

$$\nabla_{\mathbf{x}_\ell} \mathcal{J} = \sum_{i=1}^{n} \frac{\widetilde{g}_{\ell i} a_{\ell i} p_{\ell i}}{\varepsilon_\ell \varepsilon_i} (\mathbf{x}_\ell - \mathbf{x}_i) + \sum_{i=1}^{n} b_i (\mathbf{x}_i - \mathbf{x}_{k_i}) \delta_{\ell k_i} - b_\ell (\mathbf{x}_\ell - \mathbf{x}_{k_\ell}). \tag{2.3.29}$$

The proof is completed by relabelling indices. □

Remark 2.3.10. For computational purposes, we can further simplify (2.3.26) by encoding the k-nearest neighbor information as follows. Let $C = (c_{ij})_{ij}$ be the $n \times n$ matrix with $c_{ij} = 1$ if node i is the k^{th}nearest neighbor of node j, $c_{ii} = -1$, and $c_{ij} = 0$ otherwise. Then the gradient given in the expression in Eq. (2.3.26) can be expressed as

$$\nabla_{\mathbf{x}_i} \mathcal{J} = \sum_{j=1}^{n} \frac{\widetilde{g}_{ij} a_{ij} p_{ij}}{\varepsilon_i \varepsilon_j} (\mathbf{x}_i - \mathbf{x}_j) + \sum_{j=1}^{n} c_{ij} b_j (\mathbf{x}_j - \mathbf{x}_{k_j}). \tag{2.3.30}$$

This is our final equation for the gradient of the loss with respect to the feature representations $\mathbf{x}_1, \ldots, \mathbf{x}_n$; since these features are the output from a neural network, the remainder of backpropagation can be done as normal.

2.3.2.1 Sparse and efficient gradient computations

We discuss here how to efficiently compute the backpropagation gradient formula (2.3.30) in a computationally efficient way in the setting of a sparse adjacency matrix A.

We use the expression (2.3.30) from Remark 2.3.10. The matrices A (adjacency matrix), P (from Lemma 2.3.8), and C (from Remark 2.3.10) require minimal additional computational overhead, as their structure and values all come from the k-NN search and weight matrix construction required in the forward pass to construct the weight matrix W. In our experiments, for example, we use $\eta(z) = \exp(-4z)$, so $P = -4W$. The matrix G is computed as described in Section 2.3.1.

It is more efficient in implementations to express (2.3.30) as sparse matrix multiplications. The first term already has this form; indeeed, if we define the matrix $H = (h_{ij})_{ij}$ by

$$h_{ij} = \frac{\widetilde{g}_{ij} a_{ij} p_{ij}}{\varepsilon_i \varepsilon_j}$$

then the first term is given by

$$\sum_{j=1}^{n} H_{ij}(\mathbf{x}_i - \mathbf{x}_j)^T = L_H X$$

which is the multiplication of the sparse graph Laplacian matrix L_H corresponding to the symmetric matrix H with the data matrix X whose rows are the feature vectors $\mathbf{x}_i^T$.

Analogously, for the second term, define the matrix $M = (m_{ij})_{ij}$ as

$$m_{ij} = c_{ij}b_j + c_{ji}b_i.$$

We then have that

$$\begin{aligned}\sum_{j=1}^{n} m_{ij}(\mathbf{x}_i - \mathbf{x}_j) &= \sum_j m_{ij}\mathbf{x}_i - \sum_j m_{ij}\mathbf{x}_j \\ &= \sum_j c_{ij}b_j(\mathbf{x}_i - \mathbf{x}_j) + \sum_j c_{ji}b_i(\mathbf{x}_i - \mathbf{x}_j).\end{aligned}$$

Now note that, for a fixed i, the column c_{ji} ($j \in \{1, ..., n\}$) is 1 when $\mathbf{x}_j$ is the kth nearest neighbor of $\mathbf{x}_i$. A node only has 1 kth nearest neighbor, so the column c_{ji} is 0 everywhere except when $j = k_i$, $c_{k_i i} = 1$. The term $c_{ii} = -1$ is irrelevant, since $\mathbf{x}_i - \mathbf{x}_i = 0$. Hence we have

$$\begin{aligned}\sum_{j=1}^{n} m_{ij}(\mathbf{x}_i - \mathbf{x}_j) &= \sum_j c_{ij}b_j(\mathbf{x}_i - \mathbf{x}_j) + b_i(\mathbf{x}_i - \mathbf{x}_{k_i}) \\ &= \sum_j b_j(\mathbf{x}_{k_j} - \mathbf{x}_j)\delta_{ik_j} + b_i(\mathbf{x}_i - \mathbf{x}_{k_i}) \\ &= -\sum_{j=1}^{n} c_{ij}b_j(\mathbf{x}_j - \mathbf{x}_{k_j}),\end{aligned}$$

where we used the definition of the matrix C again. This is exactly the negative of the second term in (2.3.30). The expression on the left is again a graph Laplacian L_M for the symmetric matrix M, so we can write

$$\sum_{j=1}^{n} c_{ij}b_j(\mathbf{x}_j - \mathbf{x}_{k_j})^T = L_M X.$$

Finally, this allow us to express (2.3.30) as

$$\nabla_X \mathcal{J}^T = (L_H - L_M)X = L_{H-M}X. \tag{2.3.31}$$

Compared to a softmax classifier, the only additional computational overhead are sparse matrices, which can be efficiently stored and manipulated.

2.4 Implementation & Numerical Experiments

This section details the practical use of the proposed GLL classifier and showcases its strong performance through a variety of numerical experiments. Section 2.4.1 explains our implementation of the GLL as a classifier on supervised and semi-supervised tasks, including batching, contrastive pretraining, and data augmentation. Section 2.4.2 visualizes the embeddings learned by a GLL on a toy example and illustrates the effect of the Tikhonov parameter τ in (2.2.15); all remaining experiments use the standard Laplace learning algorithm in (2.2.11). Section 2.4.3 considers a deliberately over-parameterized CNN on FashionMNIST to compare a GLL head to an MLP head in terms of generalization, training dynamics, and computational efficiency. Section 2.4.4 presents large-scale experiments on the CIFAR-10 [KH09] and EMNIST [CAT17] datasets at varying levels of supervision, comparing MLP, WNLL [WO21], and GLL classification heads across several deep neural network backbones; GLL-based networks consistently match or surpass the performance of MLP- and WNLL-based networks, especially at low label rates. Section 2.4.5 quantifies the computational overhead of GLL by reporting training and inference times as a function of batch size and network size. Finally, Section 2.4.6 provides our most significant empirical result: across datasets, training strategies, and attack strengths, a GLL yields consistently and substantially stronger robustness to adversarial attacks than both MLP- and WNLL-based classifiers. While our theoretical analysis applies to general graph Laplacian-based learning algorithms, we leave numerical experiments on variants such as Poisson learning and p-Laplace learning to future work.

2.4.1 Implementation

In this subsection, we present implementation details for the GLL in our experiments. Although our GLL can be plugged into any neural network as a classifier, replacing the original head in this way leads to a batch setting that differs slightly from the classical MLP-based training paradigm. In addition, to accelerate training, we adopt a contrastive-learning–based pretraining stage, after which we perform supervised training for the neural network with the GLL.

2.4.1.1 Sampling and Training on Batches

We describe a slight modification of batching needed for a GLL classifier. Recalling the notation of Section 2.2.3, we let $\mathcal{X}$ denote the set of samples in the training set, $\mathcal{L}$ be the subset of base samples with known labels, $\mathcal{U}$ the unlabeled subset. Then we consider training on a batch $\mathcal{B}$ with size B for the classic classifier (e.g. an MLP) or for our GLL classifier:

Classic Training: Since each data is classified individually, the batch is selected only from the labeled dataset $\mathcal{L}$, i.e. $\mathcal{B} \subset \mathcal{L}$. The training loss $\mathcal{J}$ is computed between the predicted labels and their ground-truth labels over all samples in $\mathcal{B}$, which are available for every element of $\mathcal{B}$ since $\mathcal{B} \subset \mathcal{L}$.

GLL Training: The data is classified by label propagation on a graph built on a batch $\mathcal{B}$. Because the GLL is a transductive classifier, it requires constructing each batch to include an additional set of labeled data - which we call the *base dataset* $\mathcal{L}_b$ - to propagate the labels to the rest of the batch. $\mathcal{L}_b$ is a subset sampled from the labeled dataset $\mathcal{L}$ with a size N_b. The remainder of the batch includes standard training samples $\mathcal{B}_l$ - whose labels we predict and compute a loss over - and (optionally) some unlabeled samples $\mathcal{B}_u$ in (e.g.) the semi-supervised setting. The batch can thus be written as

$$\mathcal{B} = \mathcal{L}_b \cup \mathcal{B}_l \cup \mathcal{B}_u, \tag{2.4.1}$$

where $\mathcal{B}_l = (\mathcal{B} \backslash \mathcal{L}_b) \cap \mathcal{L}$ and $\mathcal{B}_u = (\mathcal{B} \backslash \mathcal{L}_b) \cap \mathcal{U}$. Let $N_l = |\mathcal{B}_l|$, $N_u = |\mathcal{B}_u|$, and $N_b + N_u = B$. We further require that, after fixing the base dataset $\mathcal{L}_b$, the proportion of labeled and unlabeled samples in each batch reflects their proportion in the remaining training set. Excluding the base samples, we sample $\mathcal{B}_l$ from $\mathcal{L} \setminus \mathcal{L}_b$ and $\mathcal{B}_u$ from $\mathcal{U}$ such that

$$\frac{N_l}{N_l + N_u} = \frac{|\mathcal{L} \setminus \mathcal{L}_b|}{|\mathcal{L} \setminus \mathcal{L}_b| + |\mathcal{U}|}. \tag{2.4.2}$$

Equivalently, for a fixed batch size B and the base dataset $\mathcal{L}_b$, we choose N_l and N_u to satisfy $N_l + N_u = B$ and

$$N_l = \frac{|\mathcal{L} \setminus \mathcal{L}_b|}{|\mathcal{L} \setminus \mathcal{L}_b| + |\mathcal{U}|} B, \quad N_u = \frac{|\mathcal{U}|}{|\mathcal{L} \setminus \mathcal{L}_b| + |\mathcal{U}|} B. \tag{2.4.3}$$

In practice, we choose a suitable batch size B and base dataset size N_b such that the resulting N_l and N_u computed from (2.4.3) are both integers. For GLL, the training loss $\mathcal{J}$ is computed between the predicted labels and ground-truth labels over all samples in $\mathcal{B}_l$. Note that although the base dataset $\mathcal{L}_b$ is labeled, those labels are already used for graph-based label propagation and are therefore not included in the loss computation.

Remark 2.4.1 (Practical usage of GLL). The proposed GLL-based batching scheme leads to several useful regimes in practice:

1. **Semi-supervised learning.** In the standard semi-supervised setting, the training set decomposes as $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ with $\mathcal{L}$ labeled and $\mathcal{U}$ unlabeled. In each batch, the nodes in $\mathcal{B}_u$ are treated as unlabeled on the graph: they do not contribute directly to the loss, but they participate in graph construction and label propagation, and hence influence the predictions on $\mathcal{B}_l$. In this way the GLL can fully exploit the unlabeled data to improve classification on the labeled subset.

2. **Choice of base dataset during training.** During training, the base dataset $\mathcal{L}_b$ is selected at the beginning of each epoch, and all batches within the same epoch share

the same base set $\mathcal{L}_b$. We consider two strategies for constructing $\mathcal{L}_b$, including a simple stratified random resampling scheme that preserves the class proportions of the full labeled set $\mathcal{L}$ and an uncertainty-based selection method. The precise sampling procedures are described in Appendix 2.6.1.1.

3. **Transductive inference at test time.** At test time, the GLL still requires a base set to construct the graph, but no backpropagation through the GLL classifier is needed. We can therefore first apply the feature encoder (the network backbone preceding the GLL classifier) to all available samples, including both training and test data, using standard mini-batch processing. Based on the resulting feature representations, we then construct a single large graph and solve the graph learning problem described in Section 2.2.3 on this graph to obtain predictions for the test nodes and compute the test accuracy. Since this stage only involves linear algebra on fixed features and does not require gradients, it can be moved to the CPU to reduce GPU memory usage.

2.4.1.2 Pretraining

Due to its transductivity, GLL accuracy and learning behavior may degenerate if the feature embeddings are poor, which is often the case for an untrained network. To remedy this, we warm the model up with a pretext task that allows the feature extractor to learn basic features. Prior works [BOC23, CMB25] show that feature representations learned from contrastive learning pre-training exhibit more pronounced clustering structure, leading to improved performance for graph-based learning methods such as GLL. We follow suit and warm-up the models using the contrastive methods SimCLR [CKN20] or SupCon [KTW20], depending on the availability of labeled data. Contrasive learning and the GLL are a natural pairing, as the loss functions for the contrastive methods use a similarity function that is effectively the same as the one used in the similarity graph construction in the GLL. Warming up the models with contrastive learning will thus help the model capture features and struc-

tures similar to the GLL and accelerate learning. The contrastive methods are discussed further in Appendix 2.6.1.2. Warm-ups were also used in previous work on graph-based classification heads [WO21], where the authors alternate between 400 epochs of MLP-based training and 5 epochs of WNLL-based training in order to stabilize learning. Conversely, we warm-up with contrastive learning, and then proceed with GLL training, with no alternating stages.

In practice, we consider a mixed training target with the SimCLR loss $\mathcal{L}_{\text{sim}}$ (2.6.3) and the SupCon loss $\mathcal{L}_{\text{sup}}$ (2.6.4). We only pretrain the feature encoder before the classifier in the neural network architecture since both losses do not require predicted labels. The pretraining loss is defined by:

$$\mathcal{L}_{\text{pretrain}} = \gamma\mathcal{L}_{\text{sim}} + (1-\gamma)\mathcal{L}_{\text{sup}}, \gamma \in [0,1]. \tag{2.4.4}$$

The contrastive pretraining described above is agnostic to the choice of classification head and can therefore be used both with classical MLP classifiers and with our GLL classifier. For each pretraining batch, we draw a mixture of labeled and unlabeled samples from the full training set $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$, in the same spirit as the batch construction in Section 2.4.1.1. The SimCLR loss $\mathcal{L}_{\text{sim}}$ is evaluated on all samples in the batch, regardless of whether they are labeled or unlabeled, since it does not require label information. By contrast, the SupCon loss $\mathcal{L}_{\text{sup}}$ is computed only on those samples in the pretraining batch that have labels, using their ground-truth class assignments. In this way, the pretraining stage can exploit all available data while still benefiting from supervised signal when labels are present, and the resulting encoder can be paired with any downstream classifier, including GLL.

Remark 2.4.2 (Choice of mixing parameter γ). In practice, we choose the mixing parameter γ in $\mathcal{L}_{\text{pretrain}}$ (2.4.4) from the discrete candidate set $\{0.01, 0.25, 0.5, 0.75, 0.99\}$. For each candidate value, we perform the contrastive pretraining stage to obtain a feature encoder and then feed the resulting feature vectors into a parameter-free graph learning classifier. We compute the resulting classification accuracy on the labeled subset $\mathcal{L} \subset \mathcal{X}$ of the training

data and select the value of γ that achieves the highest accuracy on this labeled subset.

2.4.1.3 Data Augmentation

Data augmentation is applied throughout both the contrastive pretraining stage without a classification head and the subsequent supervised training of the full network with different classifiers. This follows the standard practice in contrastive representation learning [CKN20, CKS20, KTW20], where multiple views of the same input are generated via stochastic transformations to form positive pairs. During evaluation on all benchmarks, we disable data augmentation and feed the original inputs to the encoder and classifier.

Concretely, at each training step we first sample a mini-batch of images from the training set and then apply a stochastic augmentation pipeline to each image. During contrastive pretraining, every sample in the batch is independently transformed twice to produce two augmented views, which are then used as positive pairs for the SimCLR and SupCon losses. In the fully supervised training phase, each data sample is augmented once on-the-fly to increase the diversity of training examples.

The augmentation pipeline itself is based on a RandAugment-style [CZS19] policy: for each image, a small number of transformations is drawn at random from a pool that includes common photometric and geometric operations, such as autocontrast, histogram equalization, brightness/contrast adjustments, rotations, shears, translations, and solarization, followed by a random cutout operation that masks out a square region of the image. For grayscale datasets we use a slightly modified variant of this policy that is tailored to single-channel images, e.g., removing color-only operations and using grayscale cutout. The full list of transformations and the exact sampling procedure are provided in Appendix 2.6.1.3. RandAugment is applied to all samples in the batch, including those belonging to the base dataset $\mathcal{L}_b$.

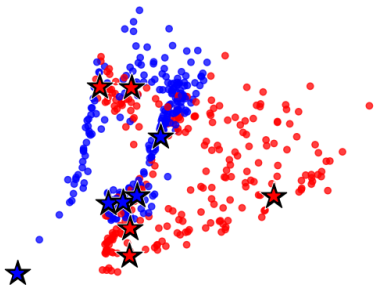


Figure 2.3: The initial embedding of the two moons dataset from the random seed. The blue and red correspond to the different classes, with the starred samples corresponding to the base samples for the respective classes. [BCH24, Figure 3]

2.4.2 Understanding Embeddings and the Effect of Tikhonov Regularization via Two Moons

In this subsection, we highlight the differences between the learned embeddings of neural networks when trained with a GLL compared to a softmax layer. Moreover, we demonstrate the effect on the embedding by modifying the decay parameter, τ, from Section 2.2.4.

The softmax classifier uses no relational information and thus will create an embedding that leads to a straightforward separation based on the features alone. The GLL, using relational information, will learn to separate the data such that the generated similarity graph will be nearly disjoint by class (a totally disjoint graph can prevent learning). Therefore, we may expect the GLL to produce a more relaxed embedding, as a disjoint similarity graph does not necessarily require the data to be separated to a probability simplex. Increasing the Tikhonov regularization, τ, inhibits label propagation and weakens predictions, meaning higher confidence predictions and lower loss will require a similarity graph that is even more separated. Specifically, increasing τ will increase the learning rate and convergence speed of the process and should lead to more locally dense learned embeddings. We give a brief example of these effects in two dimensions with the simple toy dataset, two moons [PVG11].

In this experiment, we perform supervised classification on a two-dimensional generated dataset of 500 samples from the two moons distribution with a noise of 0.1 and random state of one. The network was trained with a stochastic gradient descent optimizer with a learning rate of 0.01.

The architecture backbone for this experiment is a simple multi-layer perceptron neural network with three fully connected (FC) layers and two ReLU activation layers. The hidden layer has a size of 64 and the output size is two for two dimensional visualization purposes. We choose an initialization for the neural network such that the initial embedding has sufficiently mixed samples (Figure 2.3). As the dataset is simplistic, there will be no projection head in this comparison, meaning the only difference between the standard MLP model and the GLL model will be purely the softmax and GLL classifiers.

We arbitrarily fix the first ten samples, five from each class, to act as the base samples which propagate the labels. As the dataset is small, we load all the samples in each batch so there is no concern about partial class representation. To make the results more interpretable, we fix the ε_k term, from the similarity equation, Equation 2.2.1, to be fixed at one.

The embeddings for this ablation study are shown in Figure 2.4. We compare five values of τ: 0, 0.001, 0.01, 0.1, and 0.5. The figure shows the learned embedding of the dataset from the neural network across lengthening training epochs.

We note that this regime is deliberately constrained to two dimensions, meaning that the network may struggle to 'move' the embeddings from one class beyond the other class's base samples, as approaching the opposing base samples will increase the loss. The seed for these samples was chosen to be difficult and we note that the problem is easier in a higher dimensional space. Despite this, after sufficient training, the model successfully separates the data.

Examining the results, the baseline MLP encoder trained with the softmax layer simply separated the data approximately along a line. However, the same encoder architecture

trained with the GLL will recover the original two moons shape, implying that the two moons shape induces a strongly separated k-nearest neighbors similarity graph.

The addition of the τ decay parameter will encourage a tighter clustering of the classes around the base labels. For low values of τ, we still see initial convergence to the two moons before diverging into tighter clusters around the base samples. If the τ value is large, the divergence is more immediate and is seen splintering one of the classes into multiple clusters localized about different base samples. This is because increasing τ acts as an exponential dampening term which localizes the influence of the base samples.

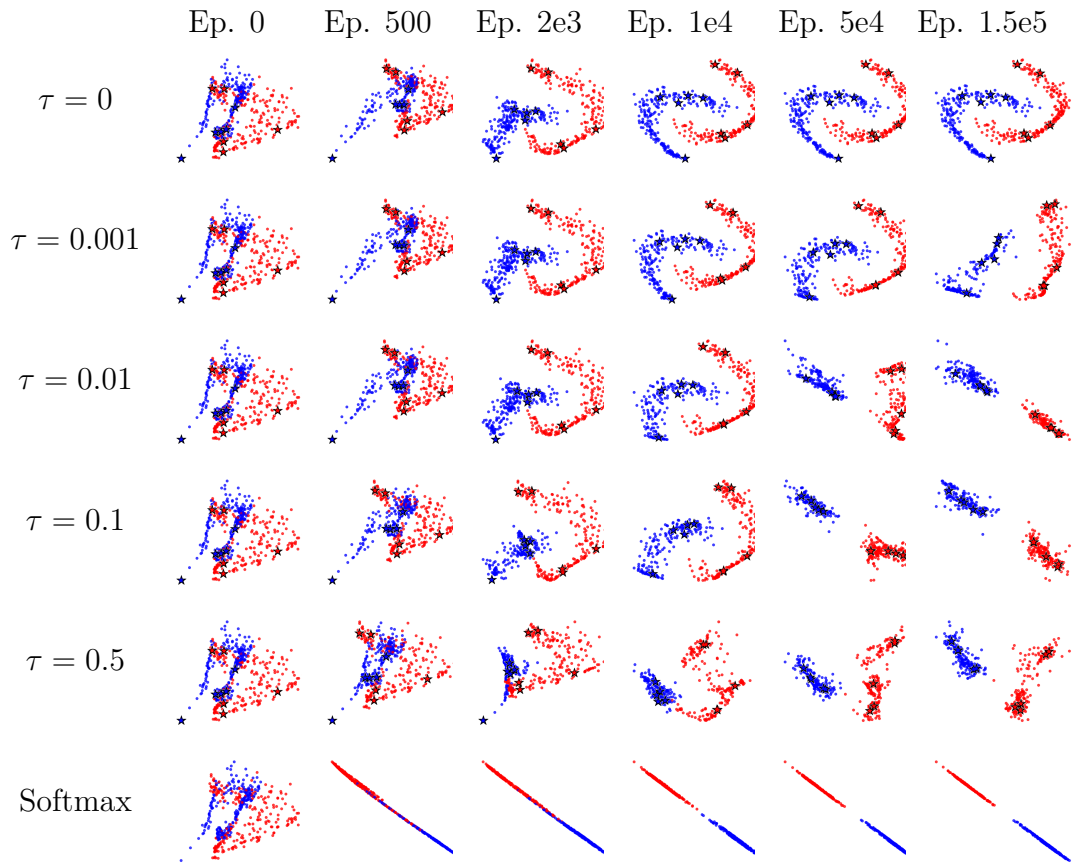


Figure 2.4: Visualization of effect of τ on spatial embeddings. Each row corresponds to a graph learning model with a specified value of τ or a softmax classification head. The columns correspond to the epoch of the training. With low values of τ, the model separates the data by class and resembles the original two moons shape. Increasing τ forces tighter clustering and localization. The softmax layer simply learns a linear separation with no regard to relational information. Further training with the softmax layer increases the scale, which is not seen without the axes. [BCH24, Figure 4]

Structure	Layer	Details	Feature Size
Input Image	-	From FashionMNIST Dataset	$28 \times 28 \times 1$
Feature Encoder	Conv1	$1 \to 64$, kernel size: 3, padding: 1	$28 \times 28 \times 64$
	Conv2	$64 \to 128$, kernel size: 3, padding: 1	$28 \times 28 \times 128$
	MaxPool	kernel size: 2, stride: 2	$14 \times 14 \times 128$
	Conv3	$128 \to 256$, kernel size: 3, padding: 1	$14 \times 14 \times 256$
	MaxPool	kernel size: 2, stride: 2	$7 \times 7 \times 256$
	FC	$256 \times 7 \times 7 \to 128$	128
MLP Classifier	FC1	$128 \to 1024$	1024
	FC2	$1024 \to 10$	10
	Softmax	dim: 1	10

Table 2.1: Network structure used in the FashionMNIST experiments (Section 2.4.3). The feature encoder is deliberately over-parameterized to create a challenging training scenario, and the model contains 1,975,424 parameters in the feature encoder and 142,346 parameters in the MLP classifier. [BCH24, Table 1]

2.4.3 Toy Example: Over-parameterized CNN on FashionMNIST

In this subsection, we focus on a toy example on the FashionMNIST dataset to compare our GLL classifier with a MLP classifier. The goal is to study a setting where the backbone network is intentionally over-parameterized and challenging to optimize, so that the effect of replacing the MLP classifier by GLL becomes more evident.

2.4.3.1 Experimental setup.

We use the FashionMNIST dataset [XRV17] and construct a simple convolutional neural network as the feature encoder. The architecture is summarized in Table 2.1. The feature encoder consists of three convolutional layers followed by max-pooling layers and a fully connected (FC) layer. Each convolutional and FC layer is followed by a ReLU activation function, which is omitted from the table for brevity. The resulting encoder is purposely over-parameterized relative to FashionMNIST, and we will see that it is difficult to train effectively when combined with a standard MLP classifier.

The MLP classifier consists of two fully connected layers followed by a softmax layer, as shown in Table 2.1. To compare with our GLL classifier, we keep the feature encoder architecture fixed and replace the MLP classifier by GLL, which has no trainable parameters. In all cases, the final outputs are used to compute the cross-entropy loss with the ground-truth labels. All experiments in this subsection are fully supervised and use the entire FashionMNIST training set.

2.4.3.2 Training strategies.

We consider four training strategies on FashionMNIST, all sharing the same feature encoder:

- **MLP only (100 MLP).** We train the network with the MLP classifier from scratch for 100 epochs. During this process, the encoder output is also passed to a GLL classifier in parallel. Since GLL has no trainable parameters, this does not affect the training dynamics. Throughout MLP training, we use the GLL outputs to compute loss (without backpropagation) and test error, allowing a direct comparison between MLP and GLL under the same encoder.
- **GLL-0 (50 GLL).** We train the network from scratch using GLL as the classifier for 50 epochs. In the plots we refer to this strategy as GLL-0.
- **GLL-50 (50 MLP + 50 GLL).** We first train the network with the MLP classifier for 50 epochs. Starting from epoch 51, we switch the classifier to GLL and continue training for another 50 epochs, using the gradients induced by the GLL loss.
- **GLL-75 (75 MLP + 25 GLL).** We first train with the MLP classifier for 75 epochs, and then switch to GLL for the remaining 25 epochs.

Dataset	Trial	Epochs	Test error (%)
FashionMNIST	MLP only (MLP)	100 MLP	45.42
FashionMNIST	MLP only (GLL)	100 MLP	14.66
FashionMNIST	GLL only (GLL-0)	50 GLL	**8.90**
FashionMNIST	GLL-50	50 MLP + 50 GLL	**8.68**
FashionMNIST	GLL-75	75 MLP + 25 GLL	**8.91**

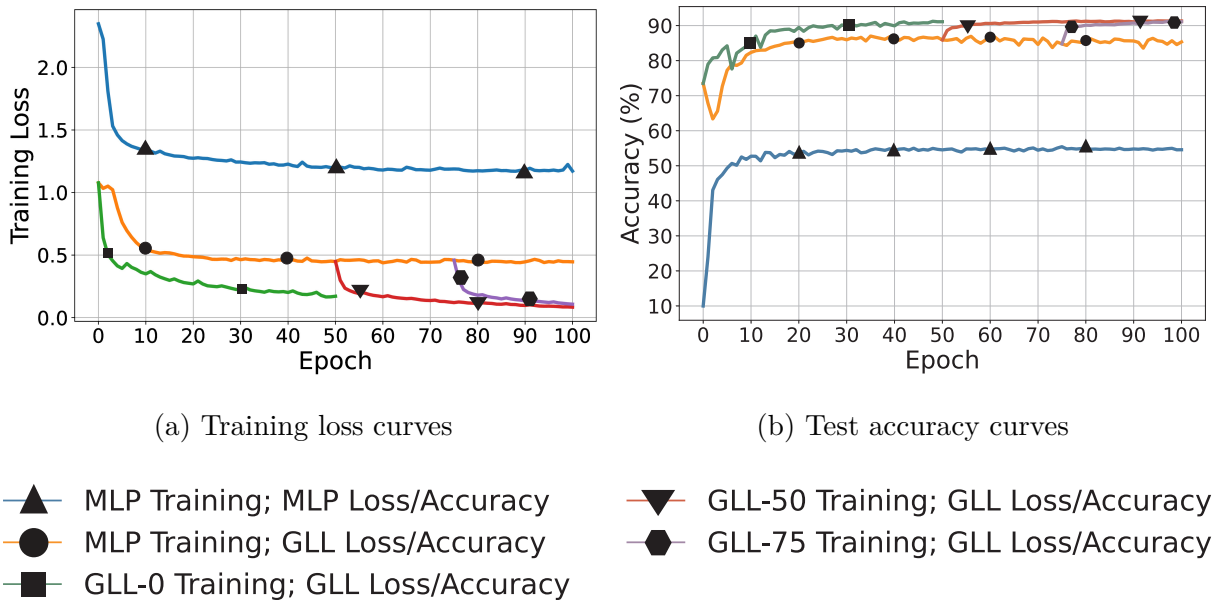
Table 2.2: Test classification error rates (%, lower is better) on FashionMNIST for the customized network in Table 2.1 under different training strategies. For the "MLP only" strategy we report both the error of the MLP classifier and that of a GLL classifier evaluated on the same encoder (GLL). For the remaining strategies, GLL is used as the classifier during training and evaluation. [BCH24, Table 2]

2.4.3.3 Test error comparison.

Table 2.2 reports the final test classification errors in %. For the MLP-only strategy, we report the test error of the MLP classifier, which is used for training, and the test error of the GLL classifier evaluated on the same encoder. For the remaining strategies, we report the test error of the GLL classifier at the end of training.

Figure 2.5 shows the evolution of the training loss and test accuracy over epochs. Figure 2.5a presents the training loss curves for all four strategies, while Figure 2.5b presents the corresponding test accuracies. The curves for GLL-50 and GLL-75 start at epochs 50 and 75, respectively, since these strategies switch to GLL after an initial MLP training phase.

From Table 2.2, the MLP classifier trained from scratch performs poorly: after 100 epochs it still incurs a test error of 45.42%, even though the GLL classifier reaches a much lower test error of 14.66% when evaluated on the same encoder. This suggests that the bottleneck is not the representation learned by the encoder, but rather the optimization of the over-parameterized MLP classifier. In contrast, when we replace the MLP with the parameter-free



(a) Training loss curves

(b) Test accuracy curves

Figure 2.5: Training loss and test accuracy of four training strategies on the **FashionMNIST** dataset with the customized feature encoder in Table 2.1. The strategies are: (1) MLP trained from scratch, (2) GLL trained from scratch (GLL-0), (3) GLL-50 (GLL after 50 epochs of MLP training), and (4) GLL-75 (GLL after 75 epochs of MLP training). For the MLP-only strategy, both the MLP and GLL losses and accuracies are recorded during training; for the remaining strategies, only the GLL classifier is used. [BCH24, Figure 5]

GLL classifier and train the same backbone (GLL-0), the test error decreases to 8.90%.

The mixed strategies GLL-50 and GLL-75 behave similarly. Although the MLP training tends to stagnate, switching to GLL mid-training yields stable decreases in the loss and further improvements in test error, reaching 8.68% and 8.91%, respectively. This indicates that once the encoder has learned a reasonable representation, GLL can immediately exploit it more effectively than the MLP classifier, and can also continue to drive smooth optimization of the encoder.

We summarize the main observations from this toy example:

1. *Error under a fixed encoder.* Even when the encoder is trained only with the MLP

classifier, the GLL achieves substantially lower loss and lower test error than the MLP on the same features. Simply replacing the MLP classifier by GLL can therefore yield a large performance gain without modifying the encoder architecture.

2. *Robustness to over-parameterization of the classifier.* The over-parameterized MLP classifier is difficult to optimize and can suffer from issues such as vanishing gradients or dying ReLUs, leading to training stagnation. The GLL classifier, having no trainable parameters, avoids these optimization pathologies and works well even when the backbone is not carefully designed.

3. *Smooth training dynamics.* When training starts with an MLP classifier and then switches to GLL (GLL-50 and GLL-75), the loss immediately resumes decreasing and the test accuracy improves further. This shows that GLL can effectively unlock stalled training and provide a smoother optimization trajectory for the shared encoder.

2.4.4 Large Scale Experiments

Network	Encoder	MLP Head	Total
VGG11	9,223,232	332,778	9,556,010
VGG13	9,407,936	332,778	9,740,714
ResNet20	271,824	16,938	288,762
ResNet32	466,256	16,938	483,194
ResNet44	660,688	16,938	677,626
ResNet56	855,120	16,938	872,058
ResNet110	1,730,064	16,938	1,747,002
ResNet18	11,168,832	332,778	11,501,610
PreActResNet18	11,171,146	332,778	11,503,924

Table 2.3: Parameter counts by network [BCH24, Table 3]

This section presents large scale experiments that compare three interchangeable classifiers under the same backbone feature encoder: (1) a two-layer MLP with softmax, (2)

Network	Whole MLP	Whole WNLL	Whole GLL	20% MLP	20% WNLL	20% GLL	2% MLP	2% WNLL	2% GLL
VGG11	7.77 ± 0.22	7.71 ± 0.10	**7.48 ± 0.19**	15.16 ± 0.58	14.74 ± 0.27	**13.14 ± 0.09**	34.76 ± 1.38	34.97 ± 1.57	**29.59 ± 0.12**
VGG13	6.16 ± 0.31	6.20 ± 0.23	**5.80 ± 0.03**	12.68 ± 0.48	12.27 ± 0.19	**11.03 ± 0.04**	31.61 ± 1.17	31.31 ± 0.49	**26.90 ± 0.18**
ResNet20	9.25 ± 0.96	7.99 ± 0.33	**7.49 ± 0.24**	15.60 ± 0.74	14.76 ± 0.68	**14.00 ± 0.31**	33.35 ± 1.81	32.04 ± 0.58	**31.25 ± 0.19**
ResNet32	8.45 ± 0.96	7.26 ± 0.29	**6.61 ± 0.15**	14.78 ± 0.58	13.63 ± 0.24	**13.04 ± 0.13**	33.93 ± 1.76	**30.82 ± 0.79**	32.96 ± 1.44
ResNet44	8.07 ± 0.78	6.57 ± 0.11	**6.12 ± 0.12**	14.38 ± 0.82	13.23 ± 0.33	**12.51 ± 0.34**	33.76 ± 2.07	31.85 ± 1.46	**31.69 ± 0.70**
ResNet56	7.69 ± 0.61	6.17 ± 0.31	**5.62 ± 0.07**	13.92 ± 0.74	12.85 ± 0.35	**12.02 ± 0.09**	32.28 ± 1.40	**30.22 ± 1.40**	30.37 ± 0.04
ResNet110	6.89 ± 0.46	5.71 ± 0.18	**4.62 ± 0.01**	14.11 ± 1.21	**12.07 ± 0.32**	12.15 ± 0.11	31.67 ± 1.72	29.67 ± 0.93	**28.73 ± 0.70**
ResNet18	5.44 ± 0.28	5.16 ± 0.21	**4.57 ± 0.13**	11.51 ± 0.44	10.53 ± 0.18	**9.44 ± 0.06**	30.27 ± 1.17	28.83 ± 0.76	**25.05 ± 0.35**
PreResNet18	5.42 ± 0.32	5.02 ± 0.18	**4.48 ± 0.01**	11.37 ± 0.46	10.94 ± 0.32	**9.89 ± 0.01**	29.75 ± 1.05	29.24 ± 1.31	**25.23 ± 0.07**

Table 2.4: CIFAR-10 test error rates (%, lower is better) comparing MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (50000 images), "20%" (10000), and "2%" (1000). Entries report mean ± one standard deviation of the test error. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Network parameter counts follow Table 2.3. [BCH24, Table 4]

WNLL [WO21], and (3) the proposed GLL. We chose (1) as a comparison point because GLL is intended to be a drop-in replacement for the standard MLP-based classification head, while (2) serves as a good intermediary comparison between MLP and GLL because it uses a graph-based classifier in testing. However, WNLL differs significantly from GLL in training. First, it uses an alternating two-stage training protocol, where Stage 1 consists of normal MLP training, and Stage 2 is WNLL training. During Stage 2, the graph classification head computes a loss, but backpropagation is handled by approximating the gradients with a linear layer. Second, Stage 1 lasts for 400 epochs at a time, while Stage 2 lasts for 5 epochs at a time. Finally, Stage 2 freezes the parameters of all but two layers of the network. In this sense, WNLL is closer to MLP in training than GLL.

As discussed in Section 2.4.1, all models are first warmed up with contrastive pretraining on the backbone without any classification head. Then, we proceed with supervised end-to-

Network	Whole MLP	Whole WNLL	Whole GLL	20% MLP	20% WNLL	20% GLL	2% MLP	2% WNLL	2% GLL
VGG11	**10.72 ± 0.14**	12.18 ± 0.79	10.76 ± 0.09	12.06 ± 0.13	13.46 ± 0.69	**11.96 ± 0.08**	15.86 ± 0.35	16.64 ± 0.78	**14.98 ± 0.13**
VGG13	**10.50 ± 0.06**	10.73 ± 0.15	10.60 ± 0.08	**11.85 ± 0.08**	15.43 ± 1.15	11.94 ± 0.10	15.91 ± 0.38	16.48 ± 0.82	**15.08 ± 0.20**
ResNet20	**9.72 ± 0.07**	10.77 ± 0.29	10.34 ± 0.05	**11.45 ± 0.06**	12.33 ± 0.21	11.60 ± 0.07	15.79 ± 0.33	17.10 ± 1.01	**14.63 ± 0.12**
ResNet32	**9.73 ± 0.06**	10.78 ± 0.37	10.15 ± 0.14	**11.41 ± 0.05**	12.26 ± 0.38	11.54 ± 0.11	15.85 ± 0.46	17.52 ± 0.90	**14.30 ± 0.08**
ResNet44	**9.92 ± 0.06**	11.30 ± 0.36	10.10 ± 0.09	11.55 ± 0.16	12.04 ± 0.28	**11.43 ± 0.14**	15.87 ± 0.35	17.81 ± 0.98	**14.98 ± 0.18**
ResNet56	9.98 ± 0.12	10.80 ± 0.44	**9.90 ± 0.12**	11.76 ± 0.14	12.24 ± 0.29	**11.60 ± 0.08**	15.74 ± 0.30	17.81 ± 0.97	**14.61 ± 0.21**
ResNet110	10.47 ± 0.09	10.74 ± 0.24	**10.19 ± 0.11**	**11.83 ± 0.12**	13.49 ± 0.54	11.99 ± 0.07	15.86 ± 0.44	16.81 ± 0.89	**14.34 ± 0.19**
ResNet18	10.57 ± 0.15	11.04 ± 0.18	**10.32 ± 0.12**	11.76 ± 0.10	12.57 ± 0.43	**11.66 ± 0.07**	16.21 ± 0.11	17.66 ± 0.83	**14.75 ± 0.09**
PreResNet18	10.57 ± 0.09	10.79 ± 0.17	**10.37 ± 0.06**	11.80 ± 0.12	12.24 ± 0.14	**11.68 ± 0.11**	15.72 ± 0.40	16.77 ± 0.64	**14.71 ± 0.13**

Table 2.5: EMNIST test error rates (%, lower is better) comparing MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (112800 images), "20%" (22560), and "2%" (2256). "Whole" uses the full standard training set. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Entries report mean ± one standard deviation of the test error. Network parameter counts follow Table 2.3. [BCH24, Table 5]

end training after attaching one of the three heads. We report the test error on CIFAR-10 and EMNIST-Balanced [CAT17] datasets. EMNIST extends MNIST by adding handwritten upper and lowercase letters. The Balanced version contains 131,600 samples and 47 classes[4]. Following a similar experimental protocol as prior work [WO21], we evaluate multiple feature encoders including variants of VGG [SZ14], ResNet [HZR16a], and PreActResNet [HZR16b] and three label fractions: full training set, 20 percent, and 2 percent based on class-balanced sampling, since graph-based classifiers are a standard choice in semi-supervised settings. Parameter counts for these networks are listed in Table 2.3.

For CIFAR-10, we train for 1000 epochs. Recalling the definitions from Section 2.4.1, we use a batch of 1500 points in training and testing, which includes 250 base labeled points

[4]Some of the upper and lowercase letters are merged into one class because they are the same shape, for example "s" and "S".

Network	Whole MLP	Whole WNLL	Whole GLL	20% MLP	20% WNLL	20% GLL	2% MLP	2% WNLL	2% GLL
VGG11	**35.88 ± 0.48**	36.92 ± 1.15	35.91 ± 0.28	38.51 ± 0.42	**36.57 ± 0.55**	38.10 ± 0.18	42.91 ± 0.76	45.05 ± 0.33	**42.14 ± 0.33**
VGG13	**34.88 ± 0.32**	36.05 ± 0.37	35.38 ± 0.34	**37.90 ± 0.23**	38.68 ± 0.74	38.11 ± 0.33	44.51 ± 1.48	45.28 ± 0.53	**42.29 ± 0.36**
ResNet20	**31.66 ± 0.21**	35.76 ± 0.59	33.56 ± 0.38	35.80 ± 0.59	36.00 ± 0.36	**35.53 ± 0.23**	43.17 ± 1.29	46.61 ± 0.96	**41.45 ± 0.41**
ResNet32	**31.69 ± 0.29**	34.47 ± 0.81	32.87 ± 0.49	**36.09 ± 0.32**	36.72 ± 0.60	36.09 ± 0.42	43.66 ± 1.35	45.80 ± 0.62	**39.94 ± 0.33**
ResNet44	**31.84 ± 0.27**	35.61 ± 0.56	32.83 ± 0.25	37.01 ± 0.56	36.47 ± 0.44	**35.98 ± 0.57**	43.70 ± 1.44	46.68 ± 0.39	**41.51 ± 0.51**
ResNet56	32.44 ± 0.45	34.62 ± 1.07	**32.31 ± 0.46**	37.41 ± 0.41	**36.89 ± 0.72**	36.93 ± 0.38	43.99 ± 1.25	45.65 ± 0.79	**41.28 ± 0.70**
ResNet110	34.03 ± 0.53	35.03 ± 0.81	**33.22 ± 0.29**	37.48 ± 0.36	**36.77 ± 0.57**	38.25 ± 0.30	44.98 ± 1.22	43.70 ± 0.65	**40.66 ± 0.70**
ResNet18	35.51 ± 0.46	35.90 ± 0.54	**35.04 ± 0.44**	38.11 ± 0.23	**36.59 ± 0.56**	37.71 ± 0.22	44.77 ± 0.54	47.10 ± 1.95	**41.32 ± 0.19**
PreResNet18	35.70 ± 0.38	35.95 ± 0.50	**35.23 ± 0.12**	38.30 ± 0.57	38.48 ± 0.35	**38.00 ± 0.23**	44.27 ± 1.25	45.02 ± 0.80	**40.87 ± 0.64**

Table 2.6: EMNIST test error rates (%, lower is better) for MLP, WNLL, and GLL output heads across architectures, evaluated under three training-set sizes: "Whole" (112800 images), "20%" (22560), and "2%" (2256). "Whole" uses the full standard training set. Error rates are computed on a class subindex of the EMNIST (balanced) label space: although the dataset contains 47 classes in total, we evaluate only the 10 hardest classes: 0, 1, 9, F, I, L, O, f, g, and q. Entries report mean ± one standard deviation of the test error. Best results for each backbone and label rate are **bolded** and results within a standard deviation of the best result are underlined. Network parameter counts follow Table 2.3. [BCH24, Table 6]

$\mathcal{L}_b$ for GLL. The remaining 1250 points are constructed with labeled ($\mathcal{B}_l$) and unlabeled ($\mathcal{B}_u$) points according to Equation 2.4.3. This batch partition also applies to WNLL during WNLL's Stage 2 ($\mathcal{L}_b$ is referred to as "template" labels in Wang and Osher [WO21]). We use SGD with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005, with cross entropy loss. We use the standard implementations of all the neural network backbones. On EMNIST, we use similar training settings, except we train for 200 epochs and use a learning rate of 0.01, and the base sample size is increased to 470 to account for the additional number of classes. All experiments were run on a single A100 GPU. For both datasets, we use the data augmentation strategy described in Section 2.4.1.

Test error rates for CIFAR-10 and EMNIST across architectures and supervision levels are given in Tables 2.4 and 2.5. On CIFAR-10, the GLL-based models are consistently the

best performing of the three classification heads, especially as training set size decreases. This empirically validates the advantage of the relational information encoded and exploited via a GLL, as well as the precise tracking of gradients. This becomes even more essential for lower label rates.

On EMNIST, GLL is better than WNLL across label rates and MLP for low label rates, and on par with MLP in the other training settings. As expected, the GLL-based networks see the largest advantage in the semi-supervsied setting (2%).

During our experiments on EMNIST, we found that the majority of the classes were easily classified by all methods, while a handful of other classes were consistently more challenging. By isolating results on only these trickier classes, we can better elucidate differences between models. To this end, we further report EMNIST test error on *only the ten hardest classes* in Table 2.6. These correspond to handwritten digits and letters that are difficult to distinguish without context[5]. Here, the advantage of GLL is more pronounced in the 2% training case.

Overall, the results show that using a GLL classification head consistently performs on par with or better than MLP or WNLL-based classifiers. The benefit of a GLL is particularly evident at low label rates. To complement these results, Section 2.4.6 shows that a GLL offers substantial robustness gains under adversarial perturbations.

2.4.5 Training and Inference Times

To emphasize the practicality of our method, we present a side-by-side comparison of training and inference times of several GLL- and MLP-based models on the MNIST dataset in Tables 2.7, 2.8, and 2.9. We train each model for 5 epochs with varying batch sizes and report the average time per batch in training and testing. For GLL, we use a base labeled set of 100 points (10 per class). All other training hyperparameters are identical throughout. To ensure a fair comparison, we also report test error alongside training and inference time. Across all

[5]Specifically, the classes “0” and “O”; “1”, “I”, and “L”; “9”, “g”, and “q”; and “F” and “f”.

Batch Size	Metric	GLL	MLP
128	Training time / batch (s)	0.0457±0.0093	0.0044±0.0006
	Testing time / batch (s)	0.0308±0.0085	0.0024±0.0027
	Testing error (%)	0.76	2.11
256	Training time / batch (s)	0.0685±0.0146	0.0061±0.0005
	Testing time / batch (s)	0.0457±0.0108	0.0039±0.0005
	Testing error (%)	1.33	1.69
512	Training time / batch (s)	0.1244±0.0280	0.0094±0.0007
	Testing time / batch (s)	0.0881±0.0206	0.0072±0.0006
	Testing error (%)	1.27	1.53
1024	Training time / batch (s)	0.2447±0.0606	0.0167±0.0012
	Testing time / batch (s)	0.1556±0.0310	0.0139±0.0009
	Testing error (%)	1.26	4.65

Table 2.7: Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is a four-layer CNN with two linear layers. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on a T4 GPU. [BCH24, Table 7]

architectures and batch sizes, GLL attains test errors that are comparable to or better than those of the corresponding MLP baselines. For the small CNN backbone, GLL consistently achieves noticeably lower error than MLP, particularly at large batch sizes. For ResNet-18 and ResNet-110, the two methods are typically within a fraction of a percentage point of each other, with GLL often slightly better and only occasionally slightly worse. Note that this setting is simplistic - with no pretraining or hyperparameter optimization - and only meant to illustrate a computational efficiency comparison between two methods with similar test performance.

GLL is naturally slower due to the additional linear solve in the forward and backward passes: depending on the architecture and batch size, the per-batch training time increases by roughly a factor of two to an order of magnitude relative to MLP, with similar trends for

Batch Size	Metric	GLL	MLP
128	Training time / batch (s)	0.1537±0.0121	0.0597±0.0083
	Testing time / batch (s)	0.0659±0.0068	0.0220±0.0026
	Testing error (%)	0.84	0.86
256	Training time / batch (s)	0.2295±0.0193	0.1157±0.0168
	Testing time / batch (s)	0.1045±0.0162	0.0426±0.0060
	Testing error (%)	0.94	1.04
512	Training time / batch (s)	0.3851±0.0362	0.2292±0.0323
	Testing time / batch (s)	0.1834±0.0236	0.0924±0.0102
	Testing error (%)	0.97	1.34
1024	Training time / batch (s)	0.7129±0.0773	0.4676±0.0734
	Testing time / batch (s)	0.3404±0.0352	0.2032±0.0340
	Testing error (%)	0.94	2.46

Table 2.8: Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is ResNet-18. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on a T4 GPU. [BCH24, Table 8]

testing time. Importantly, however, the extra cost of GLL is independent of the number of parameters in the backbone; its main overhead scales with the size of the base labeled set rather than network width or depth. Consequently, as the architecture becomes larger, from a small CNN to ResNet-18 and ResNet-110, GLL-based models progressively approach the runtime of their standard MLP-based counterparts, while retaining competitive or improved test error.

2.4.6 Adversarial Robustness

In this subsection, we demonstrate our method's superior robustness to adversarial attacks compared to the typical projection head and softmax activation layer in a neural network. Adversarial attacks seek to generate *adversarial examples* [SZS13]: input data with small

Batch Size	Metric	GLL	MLP
128	Training time / batch (s)	0.1257±0.0044	0.0471±0.0024
	Testing time / batch (s)	0.0629±0.0035	0.0224±0.0037
	Testing error (%)	1.65	2.39
256	Training time / batch (s)	0.1772±0.0065	0.0725±0.0028
	Testing time / batch (s)	0.0950±0.0103	0.0404±0.0043
	Testing error (%)	2.10	1.18
512	Training time / batch (s)	0.2950±0.0218	0.1480±0.0637
	Testing time / batch (s)	0.1605±0.0143	0.0774±0.0074
	Testing error (%)	1.14	1.13
1024	Training time / batch (s)	0.5214±0.0605	0.2802±0.0224
	Testing time / batch (s)	0.2919±0.0208	0.1498±0.0102
	Testing error (%)	1.93	1.57

Table 2.9: Comparison of GLL and MLP performance across different batch sizes on MNIST. The architecture backbone is ResNet-110. We use identical training settings for both models. For time metrics, each entry is reported as mean ± standard deviation over five epochs of training. Testing error is reported in percentage points. The experiments were run on an A100 GPU. [BCH24, Table 9]

perturbations that cause the model to misclassify the data. Many highly performant deep neural networks can be fooled by very small changes to the input, even when the change is imperceptible to a human [MMS17, GSS14]. Hence, training models robust (i.e. resistant) to such attacks is important for both security and performance. Our experiments show that in both naturally and robustly-trained models, using the graph learning layer improves adversarial robustness compared to a softmax classifier on MNIST [Yan10], FashionMNIST [XRV17], and CIFAR-10 [KH09] in both relatively shallow and deep networks. Moreover, the GLL is more robust to adversarial attacks *without sacrificing performance on natural images*.

2.4.6.1 Adversarial Attacks

We consider three types of adversarial attacks: the fast gradient sign method (FGSM) [GSS14] in the ℓ_∞ norm, the iterative fast gradient sign method (IFGSM) [KGB18] in the ℓ_∞ norm, and the Carlini-Wagner attack [CW17] in the ℓ_2 norm.

The FGSM and IFGSM attacks consider the setting where an attacker has an ℓ_∞ "budget" to perturb an image x. More formally, FGSM creates an adversarial image x' by maximizing the loss $\mathcal{J}(x', y)$ subject to a maximum allowed perturbation $\|x - x'\|_\infty \leq \varepsilon$, where ε is a hyperparameter. Thus, a larger ε is more likely to successfully fool a model, but also more likely to be detectable. We can linearize the objective function as

$$\mathcal{J}(x', y) = \mathcal{J}(x, y) + \nabla_x \mathcal{J}(x, y)(x - x')$$

which gives the optimal (in this framework) adversarial image

$$x' = x + \varepsilon \cdot \text{sign}(\nabla_x \mathcal{J}(x, y)) \tag{2.4.5}$$

It is also standard to clip the resulting adversarial example x' to be within the range of the image space (e.g. $[0, 1]$ for grayscale images) by simply reassigning x' to $\min\{1, \max\{0, x'\}\}$.

Intuitively, FGSM computes the gradient of the loss with respect to each pixel of x to determine which direction that pixel should be perturbed to maximize the loss. It should be noted that, as the name suggests, FGSM is designed to be a *fast* attack (the cost is only one call to backpropagation), rather than an optimal attack.

IFGSM simply iterates FGSM by introducing a second hyperparameter α, which controls the step size of the iterates:

$$x_{i+1} = x_i + \alpha \cdot \text{sign}(\nabla_{x_i} \mathcal{J}(x_i, y))$$

where $x_0 = x$ and $i = 0, ..., N - 1$, and so x_N is the generated adversarial example. However, we must also clip the iterates (pixel-wise) during this iteration to ensure that (1) they remain

in the ε-neighborhood of x and (2) they remain in the range of the image space (e.g. in [0,1] for a grayscale image). Thus, the IFGSM attack is:

$$x_{i+1} = \text{clip}_{x,\varepsilon}\big\{x_i + \alpha \cdot \text{sign}(\nabla_{x_i}\mathcal{J}(x_i, y))\big\} \tag{2.4.6}$$

where

$$\text{clip}_{x,\varepsilon}\{z\} = \min\big\{1, x+\varepsilon, \max\{0, x-\varepsilon, z\}\big\}$$

Note that this assumes the pixel values are in the range $[0, 1]$, but can be easily tailored to other settings. IFGSM attacks have been shown to be more effective than FGSM attacks [KGB18], however we will see in Section 2.4.6.3 that our graph learning layer is highly robust to this iterative method.

CW attacks frame the attack as an optimization problem where the goal is to find a minimal perturbation δ that changes the model's classification of an input x. Let F be the neural network, so that $F(x)$ gives a discrete probability distribution over k classes for an input x. Let $C(x) = \text{argmax}_k F(x)$ be the classification decision for x. The CW attack is a perturbation δ that solves:

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_2^2 \\ \text{s.t.} \quad & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{aligned}$$

where t is a classification such that $C(x) \neq t$. In other words, the CW attack seeks a minimal perturbation δ to apply to x that changes its classification (while remaining a valid image). Other norms on δ are possible; in this chapter we focus on the ℓ_2 norm. In practice, we choose t to be the class with the second highest probability in $F(x)$, i.e. $t = \text{argmax}_{k \neq C(x)} F(x)$

The constraint $C(x+\delta) = t$ is highly non-linear, so Carlini and Wagner instead consider objective functions g that satisfy $C(x+\delta) = t$ if and only if $g(x+\delta) \leq 0$. While there are many choices for g ([CW17], Section IV A), we will use the following in our experiments:

$$g(x') = \big(\max_{i \neq t} F(x')_i - F(x')_t\big)^+$$

where $a^+ = \max(0, a)$. It is worth noting that a common choice for g in the literature is to use the unnormalized logits from the neural network (that is, the output of a neural network just before the standard softmax layer) instead of $F(x)$. While both definitions satisfy the condition, we use the above because graph learning methods (such as Laplace learning) operate in the feature space and directly output probability distributions over the classes, so there is no notion of logits.

Using g, we can reformulate the problem as

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_2^2 + c \cdot g(x+\delta) \\ \text{s.t.} \quad & x + \delta \in [0,1]^n \end{aligned}$$

where $c > 0$ is a suitably chosen hyperparameter. To make the problem unconstrained, we can introduce a variable w and write $\delta = \frac{1}{2}(\tanh w + 1) - x$; since $-1 \leq \tanh w \leq 1$, we have that $0 \leq x + \delta \leq 1$. This gives the unconstrained objective:

$$\min_{w} \quad \|\tfrac{1}{2}(\tanh w + 1) - x\|_2^2 + c \cdot g(\tfrac{1}{2}(\tanh w + 1)) \tag{2.4.7}$$

which can be solved using standard optimization routines. We use the Adam optimizer [KB14].

2.4.6.2 Adversarial Training

To train adversarially robust networks, we use projected gradient descent (PGD) adversarial training [MMS17] on our models. Motivated by guaranteeing robustness to first-order adversaries, PGD training applies IFGSM to images with a random initial perturbation during training. Hence, the model is trained on adversarial examples. More precisely, PGD training consists of two steps each batch: first, x is randomly perturbed

$$x^* = x + U(-\varepsilon, \varepsilon),$$

where U is a uniform random perturbation. Then, IFGSM is applied

$$x_{i+1} = \text{clip}_{x,\varepsilon}\big\{x_i + \alpha \cdot \text{sign}(\nabla_{x_i} \mathcal{J}(x_i, y))\big\},$$

where $x_0 = x^*$ and the number of iterations and α are hyperparameters. This training procedure, while more expensive, has been shown to be effective at improving robustness in deep learning models compared to training on clean (i.e. natural) images [MMS17, WO21].

2.4.6.3 Adversarial Experiments

We evaluate the adversarial robustness of neural networks with the GLL classifier on the MNIST [Yan10], FashionMNIST [XRV17], and CIFAR-10 [KH09] benchmark datasets, and compare our method to the same networks trained with either a softmax classifier or the WNLL classifier [WO21]. For all datasets, we normalize the images to have mean zero and standard deviation one, and we train and run all adversarial attacks on a 11GB GeForce RTX 2080Ti GPU. We note that the goal of these experiments is not to get state-of-the-art natural accuracies, but rather to demonstrate our method's superior robustness to adversarial attacks compared to the typical MLP classifier without sacrificing performance on natural images. A summary of our experiments, highlighting a handful of our adversarial robustness results, can be found in Table 2.10, while complete results are shown in Figures 2.6, 2.7, and 2.8. Throughout, "MLP" refers to a model with a final linear layer mapping from the feature space to the logits followed by a softmax classifier, "WNLL" refers to a model with the WNLL classifier, and "GLL" refers to a model with our graph learning layer classifier. WNLL and GLL are both drop-in replacements for the MLP classifier, and applied on the feature space itself. For these experiments, we do not include any unlabeled samples $\mathcal{B}_u$ in batch training (because the setting is fully supervised), and we do not pretrain to demonstrate that GLL-based models can be trained from scratch with desireable natural and robust accuracies.

For MNIST, we use the same "small CNN" architecture as in Wang and Osher [WO21]: four convolutional layers with ReLU activations after each layer and maxpooling after the second and fourth layer, followed by three linear layers and a softmax classifier. We train for 100 epochs using the Adam optimizer [KB14] with an initial learning rate of 0.01 which decays by a factor of 0.1 every 25 epochs. For the robustly trained models, we run PGD

Dataset	Model	Natural	FGSM	IFGSM	CW
MNIST	GL Robust	0.55	**2.28**	**2.30**	**1.16**
	WNLL Robust	0.58	2.48	2.97	2.72
	MLP Robust	0.80	4.83	6.43	2.59
	GL Natural	**0.47**	3.49	3.56	1.29
	WNLL Natural	0.50	3.54	4.64	3.19
	MLP Natural	0.75	5.74	8.48	2.76
Fashion-MNIST	GL Robust	8.96	11.93	12.83	**16.19**
	WNLL Robust	8.11	**10.97**	**11.31**	18.55
	MLP Robust	7.92	13.50	14.33	19.00
	GL Natural	7.86	34.56	38.19	42.81
	WNLL Natural	**6.44**	21.46	39.33	74.59
	MLP Natural	8.35	59.19	81.70	37.99
CIFAR-10	GL Robust	9.40	**19.50**	**20.38**	**21.91**
	WNLL Robust	9.34	25.22	28.36	50.35
	MLP Robust	8.71	26.05	28.96	24.06
	GL Natural	5.71	40.95	45.59	22.77
	WNLL Natural	5.13	58.20	92.33	93.87
	MLP Natural	**5.02**	49.25	91.79	47.00

Table 2.10: Selection of adversarial results of models trained robustly and naturally with our method compared to the WNLL and MLP classifier. We report natural and adversarial error rate on FGSM, IFGSM, and CW attacks. Best error rates are in bold. For MNIST, we report $\varepsilon = 0.3$ for the FGSM and IFGSM attacks, and $\varepsilon = 0.05$ for the other datasets. We report CW results for $c = 20$. We report more complete results across a range of values for ε and c in Figures 2.6, 2.7, and 2.8. We see that GLL is significantly more robust to adversaries across datasets without sacrificing natural performance. [BCH24, Table 10]

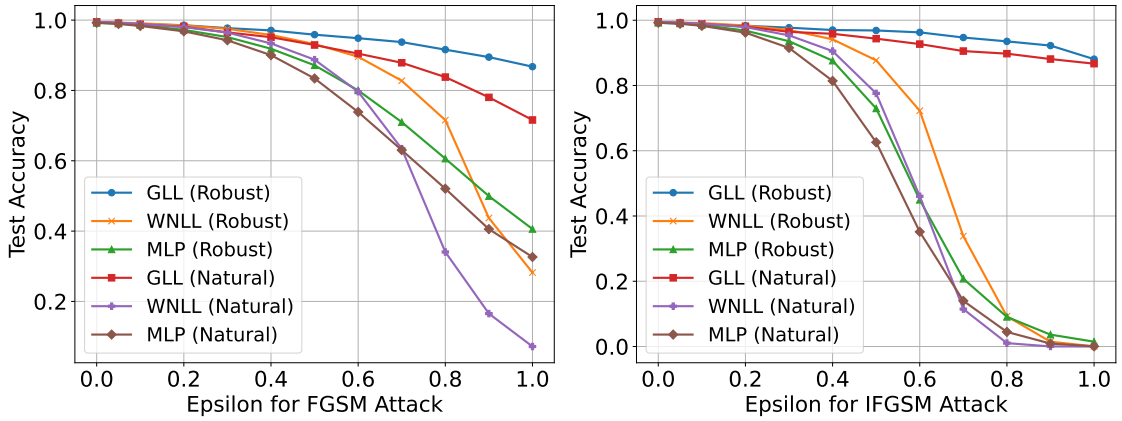
training with $\varepsilon = 0.3$ and $\alpha = 0.01$ for 5 iterations. In training, we use a batch size of 1000, and for GLL and WNLL we use 100 labeled "base" points per batch (i.e. 10 per class), which are randomly sampled each epoch.

To evaluate experimental robustness, we attack the test set with the FGSM, IFGSM, and CW attacks. When using FGSM and IFGSM on GLL, we embed the entire test set of 10000 digits along with 10000 randomly selected data points from the training set (i.e. 1000 per class) when running the attacks. We run CW in batches of 1000 test and 1000 training

points due to computational constraints; in general graph learning methods perform better with more nodes on the graph as it better approximates the underlying data manifold. We use the same testing batch sizes (full batch on FGSM and IFGSM, 1000 for CW) for the MLP and WNLL models, and the same number of base points for WNLL. We report results for FGSM and IFGSM at varying levels of ε, fixing $\alpha = 0.05$ for IFGSM and letting the number of iterations be $5 * (\varepsilon/\alpha)$, following heuristics from prior work [KGB18] to allow the adversarial example sufficient ability to reach the boundary of the ε-ball. For CW attacks, we let c vary and run 100 iterations of the Adam optimizer with a learning rate of 0.005. Results for each attack, comparing both robustly and naturally trained models with MLP and GLL classifiers, are shown in Figure 2.6.

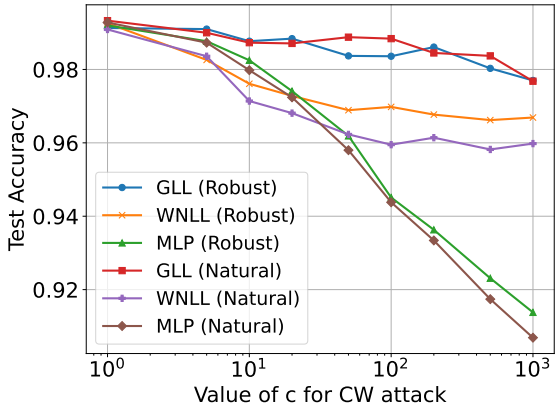
For FashionMNIST, we use ResNet-18 [HZR16a], where again we replace the final linear layer and softmax with GLL. We again use the Adam optimizer for 100 epochs, with an intial learning rate of 0.01 which decays by a factor of 0.5 every 10 epochs. For the robustly trained models, we run PGD training with $\varepsilon = 0.05$ and $\alpha = 0.01$ for 5 iterations. In training, we use a batch size of 2000, and for GLL and WNLL we use 200 labeled "base" points per batch. We attack in batches of 250 test points, with 500 training points for GLL and WNLL; this was chosen as it was the largest batch size we could use given our computational resources. The hyperparameters for the attacks are identical to those for MNIST. We report results in Figure 2.7.

Finally, for CIFAR-10, we use PreActResNet-18 [HZR16b] as our architecture backbone. We use stochastic gradient descent with momentum set to 0.9, a weight decay of 0.0005, and an intial learning rate of 0.1. We train for 150 epochs and use cosine annealing to adjust the learning rate [LH16]. For the robustly trained models, we run PGD training with $\varepsilon = 0.05$ and $\alpha = 0.01$ for 5 iterations. In training, we use a batch size of 200, and for GLL and WNLL we use 100 labeled "base" points per batch. We attack in batches of 200 test points, with 500 training points for GLL and WNLL; this was chosen as it was the largest batch size we could use given our computational resources. The hyperparameters for the attacks



(a) FGSM

(b) IFGSM

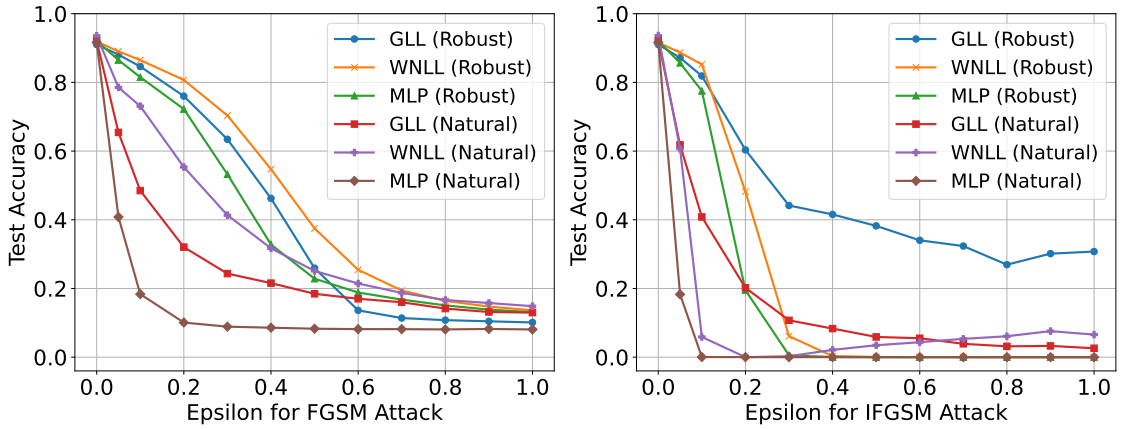


(c) CW

Figure 2.6: Adversarial robustness to FGSM, IFGSM, and CW attacks on the MNIST dataset. The backbone network is the same in all cases; the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 6]

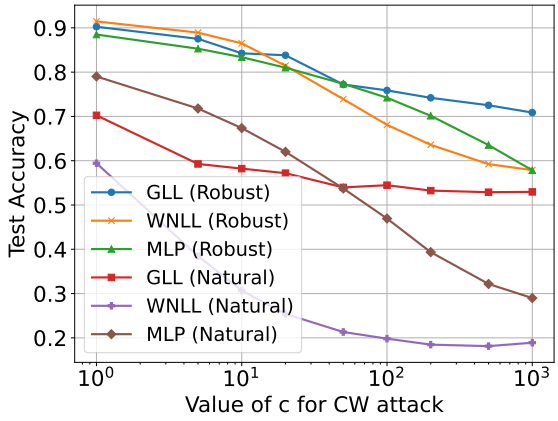
are identical to those for MNIST, except we use 50 iterations of optimization for the CW attacks. We report results in Figure 2.8.

Taken together, our results (Table 2.10 and Figures 2.6, 2.7, and 2.8) demonstrate that across datasets, training methods (both natural and robust), severity of attacks, and neural network architectures, models trained with GLL are significantly more robust to adversarial



(a) FGSM

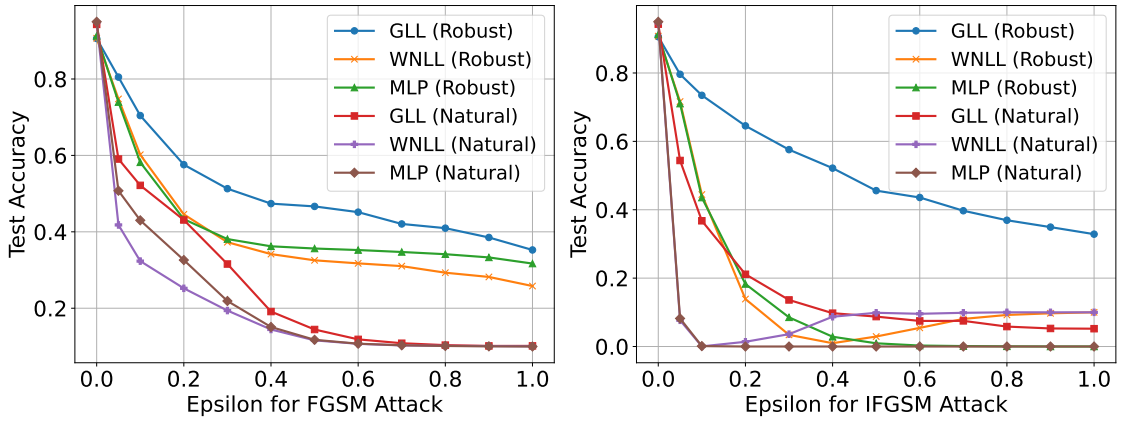
(b) IFGSM



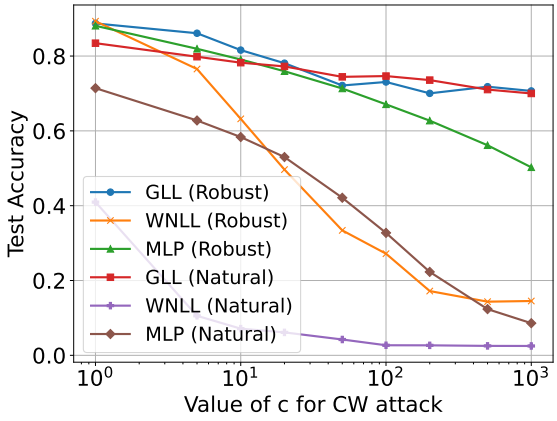
(c) CW

Figure 2.7: Adversarial robustness to FGSM, IFGSM, and CW attacks on the FashionMNIST dataset. The backbone network is the same in all cases (ResNet-18); the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 7]

attacks, without sacrificing natural accuracy. Laplace learning can be viewed as repeatedly computing a weighted average over a given node's neighbors, and this relational information may be preventing attacks from successfully perturbing the input images to fool the network compared to the MLP classifier models. Moreover, our results suggest that tracking the gradients exactly (GLL), rather than approximating them (WNLL), offers a significant



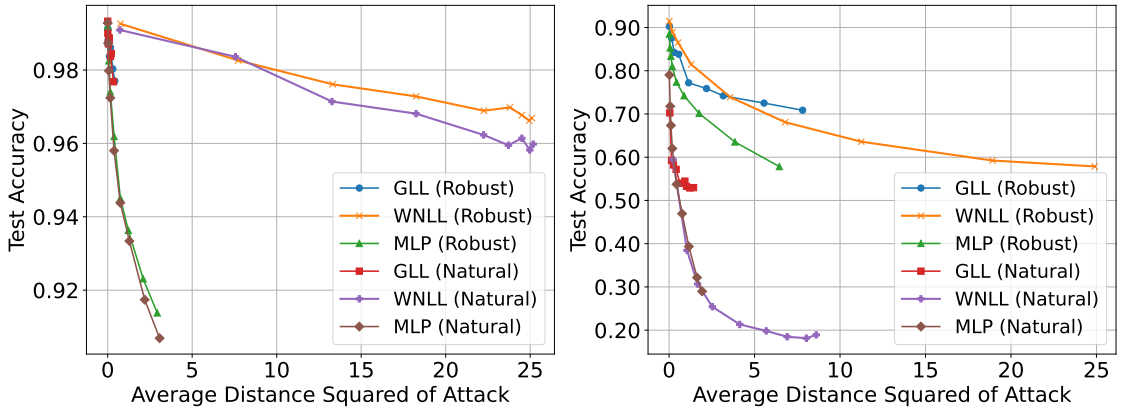
(a) FGSM (b) IFGSM



(c) CW

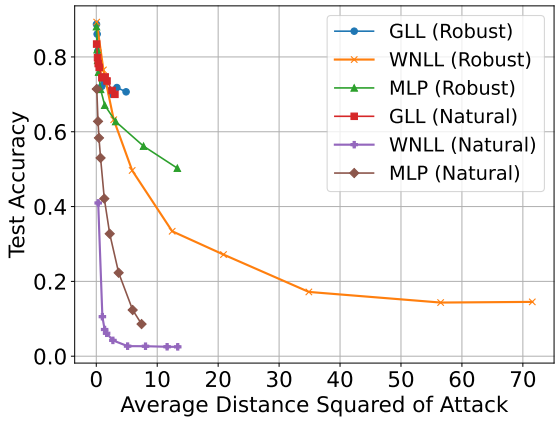
Figure 2.8: Adversarial robustness to FGSM, IFGSM, and CW attacks on the CIFAR-10 dataset. The backbone network is the same in all cases (PreActResNet-18); the only difference is the classifier (either MLP with softmax, WNLL, or our novel GLL). We report results for both robustly and naturally trained models. [BCH24, Figure 8]

increase in robustness. GLL's advantage is even more apparent for stronger attacks, whereas MLP and WNLL-based models decay much more quickly toward random guessing or even 0% accuracy. Moreover, performance of our GLL could likely be even further improved with more computational resources, as graph learning techniques tend to improve as the number of nodes (i.e. the batch size) increases.



(a) MNIST

(b) FashionMNIST



(c) CIFAR-10

Figure 2.9: Comparison of accuracy vs. average distance of attack for CW attack on all three datasets. Each point represents a different value of c. We see that GLL resists the adversary from moving the target images significantly more than the standard MLP or WNLL classifier, suggesting that graph-based classification *and* precise tracking of gradients is key for adversarial robustness. [BCH24, Figure 9]

We further investigate the improved robustness of GLL against CW attacks in Figure 2.9 (see also Appendix 2.6.2.3). This figure plots the accuracy versus the average distance (squared) that the adversary moves an image in the test set (instead of the value of c). This reveals a striking difference: models trained with the GLL classifier *resist the CW adversary*

from moving the images significantly more than the MLP or WNLL-based models, resulting in more robustness to these attacks. Interestingly, WNLL is the least resistant to these perturbations. Since WNLL uses linear layer gradients and GLL uses precise gradients, these results suggest that the exact backpropagation in GLL aids robustness to these attacks. One hypothesis is that the relational information in GLL's gradients "trap" data points in neighborhoods with points of the same class. Theoretically guaranteeing GLL's adversarial robustness is an interesting line of future work.

2.5 Conclusion

Graph-based learning is a powerful class of machine learning techniques that have been developed throughout the 21st century. As research in deep learning has exploded throughout the same time period, there has been significant interest in integrating deep learning with similarity matrix-based graph learning. This chapter gives a precise mathematical framework to do exactly that. We derived the necessary gradient backpropagation equations for general nonlinear elliptic graph Laplace equations, allowing us to fully integrate graph-based classifiers into the training of neural networks. Using this novel framework, we introduced the Graph Learning Layer, or GLL, which replaces the standard projection head and softmax classifier in classification tasks. Experimentally, we demonstrated how the GLL improves upon the standard classifier and previous graph-based layers (WNLL) in several ways. Two illustrative examples demonstrated that GLL-based networks can learn the geometry of the underlying data and improve training dynamics in overparameterized settings, respectively. In a suite of large scale experiments, we showed that networks trained with a GLL classifier have improved generalization on benchmark datasets, especially when training labels are limited. Although the improvement in natural accuracy was occasionally modest, later experiments demonstrated the GLL's significant and consistent performance gains for networks in the adversarial setting. This points toward a practical advantage of GLL in safety- and

security-critical settings. Moreover, by comparing to WNLL-based networks, our results also empirically validated the advantage of tracking gradients through graph learning *precisely*, rather than approximately.

There are many exciting avenues of future work in this area. Given the strong empirical results, theoretically guaranteeing the adversarial robustness of the GLL is an important extension of this chapter. Graphs are a common setting for active learning techniques [Set09, MB24]; the full integration of graph learning into a neural network provides a natural way to do active learning in a deep learning context. Our theoretical work also provides a way to integrate other graph learning algorithms - such as p-Laplace learning and Poisson learning - into a deep learning framework, and numerical experiments could yield similarly strong results as reported in this chapter. Many performant semi-supervised deep learning techniques utilize psuedolabels generated by graph learning methods [ITA19, SAS22]; we hope the methods presented here will lead to further improvements and insights.

2.6 Appendix

2.6.1 Training Details

This section provides a detailed discussion of various aspects of the training process. These details are implemented in our experiments (Section 2.4), especially in the fully-supervised training for comparison between the MLP and GLL classifiers (Section 2.4.3).

2.6.1.1 Sampling from Base and Unlabeled Datasets

In the GLL-based training process, our approach to constructing mini-batches differs from classical fully supervised training. As described in Section 2.4.1.1, each mini-batch $\mathcal{B}$ contains a base dataset $\mathcal{L}_b \subset \mathcal{L}$ together with additional labeled and unlabeled samples $\mathcal{B}_l$ and $\mathcal{B}_u$. For each mini-batch, in addition to the images whose labels are used to compute the loss (the

samples in $\mathcal{B}_l$), we also require base images and their associated base labels (the samples in $\mathcal{L}_b$), which act as labeled nodes for graph-based label propagation. While this base dataset can be kept fixed throughout training, we also consider two update strategies for $\mathcal{L}_b$ that can further enhance performance.

First, in the *random resampling* strategy, we update the base dataset at the beginning of each epoch by sampling a random subset $\mathcal{L}_b \subset \mathcal{L}$ with $|\mathcal{L}_b| = N_b$. To provide stable supervision for label propagation, $\mathcal{L}_b$ is chosen so that the class proportions in $\mathcal{L}_b$ match the class proportions in the full labeled set $\mathcal{L}$, for example by stratified sampling.

Second, in an *uncertainty-based* strategy, we explicitly score labeled samples by how informative they would be if selected into $\mathcal{L}_b$. In the graph Laplace learning framework, the base dataset $\mathcal{L}_b$ represents the labeled nodes and can be viewed as the sources in a label propagation process. An ideal base dataset should be relatively dispersed in the graph and lie near decision boundaries between clusters, so that it provides informative supervision to the unlabeled nodes.

Let $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ denote the full training set, where $\mathcal{L}$ is the labeled subset and $\mathcal{U}$ is the unlabeled subset, and let N_b be the desired size of the base dataset. Assume that at a given epoch, the GLL prediction for a labeled sample $\mathbf{x}_j \in \mathcal{L} \setminus \mathcal{L}_b$ is

$$\mathbf{u}_j = \left(u_j^1, u_j^2, \ldots, u_j^C\right),$$

where C is the number of classes and u_j^c is the predicted confidence for class c. We can define two uncertainty-based scores for $\mathbf{u}_j$:

$$\text{Entropy score} = -\sum_{c=1}^{C} u_j^c \log(u_j^c), \tag{2.6.1}$$

$$\text{L2 score} = 1 - \|\mathbf{u}_j\|_2. \tag{2.6.2}$$

Both scores are larger when the prediction $\mathbf{u}_j$ is more uncertain. In this second strategy, we select the new base dataset $\mathcal{L}_b$ as the N_b labeled samples in $\mathcal{L}$ with the highest scores, according to either (2.6.1) or (2.6.2).

However, there is a risk that the highest-scoring samples may cluster in a small region of the graph, providing redundant information. A more effective variant is to employ the LocalMax batch active learning method [CCT23, CMB24]. In this approach, nodes in the graph are first assigned uncertainty scores, and then a subset is selected that both has high scores and satisfies a local maximum condition in the graph. This discourages the selection of nodes that are too close to each other and leads to a more diverse and informative base dataset $\mathcal{L}_b$ for GLL.

2.6.1.2 Contrastive Learning and Warm Up

Prior to GLL-based fully supervised training for the CIFAR-10 dataset, we initialize the network's feature encoder using pre-trained weights from SimCLR [CKN20, CKS20]. SimCLR is a framework for self-supervised contrastive learning of visual representations. For a neural network f and a minibatch of m samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, each $\mathbf{x}_k$ is augmented into pairs $\widetilde{\mathbf{x}}_{2k-1}, \widetilde{\mathbf{x}}_{2k}$. These are processed through the network: $\mathbf{z}_{2k-1} = f(\widetilde{\mathbf{x}}_{2k-1}), \mathbf{z}_{2k} = f(\widetilde{\mathbf{x}}_{2k})$. The SimCLR loss is defined as:

$$\begin{aligned} \mathcal{L}_{\text{sim}} &= \frac{1}{2m} \sum_{k=1}^{m} [\ell(2k, 2k-1) + \ell(2k-1, 2k)], \\ \ell(i, j) &= -\log \frac{\exp(g(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k\neq i}^{2m} \exp(g(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \end{aligned} \tag{2.6.3}$$

where τ is a temperature parameter, and $g(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\top \mathbf{z}_j / (\|\mathbf{z}_i\| \|\mathbf{z}_j\|)$ is the cosine similarity. Essentially, SimCLR teaches the network to increase the similarity between embedded pairs and reduce the similarity between the other embedded pairs. It is worth noting that the cosine similarity used in the SimCLR loss (2.6.3) closely resembles the similarity measure we employ when constructing graphs in graph Laplace learning. This similarity suggests that the feature vectors generated by a network trained with SimCLR already possess a favorable clustering structure. Consequently, utilizing SimCLR to pretrain the feature encoder in our network can significantly reduce the difficulty of subsequent training stages.

In the supervised setting, SupCon [KTW20] can be used instead of SimCLR. With full access to the underlying labels, the goal is now to maximize the similarity between all embedded pairs within the same class and minimize the similarity of samples across classes. The loss function is

$$\mathcal{L}_{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)} \tag{2.6.4}$$

where $P(i) \equiv \left\{p \in A(i) : \widetilde{\boldsymbol{y}}_p = \widetilde{\boldsymbol{y}}_i\right\}$ is the set of indices for all samples that belong to the same class as sample i (excluding itself) and $A(i)$ is the set containing all indices except i.

We employed augmentation techniques in both the SimCLR pre-training process and the subsequent fully supervised training. For data augmentation in both stages, we utilized the strong augmentation method defined in the SupContrast GitHub Repository[6]. The use of data augmentation is helpful for two main reasons. Firstly, in contrastive learning, augmentation creates diverse views of the same instance, which is fundamental to learning robust and invariant representations. This process helps the model to focus on essential features while disregarding irrelevant variations. Secondly, for general training tasks, augmentation enhances the model's ability to generalize by exposing it to a wider variety of data representations, thereby reducing overfitting and improving performance on unseen data.

For the purposes of warming up the GLL, either SimCLR or SupCon can be used depending on the framework. In our work, we have even seen success in balancing a combination of the two methods to utilize the labels for a strong embedding while not overfitting too strongly. The pretraining loss is taken to be the same mixed objective as in (2.4.4),

$$\mathcal{L}_{\text{pretrain}} = \gamma \mathcal{L}_{\text{sim}} + (1 - \gamma) \mathcal{L}_{\text{sup}}$$

for some $\gamma \in [0, 1]$.

[6] `https://github.com/HobbitLong/SupContrast/tree/master`

2.6.1.3 Data Augmentation Details

We describe here the data augmentation policies used during both contrastive pretraining and supervised training.

Augmentation pool for color images. For color image datasets, we adopt a policy similarly to the RandAugment [CZS19]. Let $\mathcal{T}$ denote a pool of primitive transformations including

- Photometric operations: autocontrast, histogram equalization, brightness, contrast, color, sharpness, posterization, solarization;
- Geometric operations: rotations, horizontal shears and vertical shears, horizontal translations and vertical translations;
- The identity transform.

For each image in a mini-batch, we randomly sample a fixed number of transformations from $\mathcal{T}$ with replacement and apply them sequentially. Each selected transform is assigned a magnitude drawn uniformly from a transform-specific range. For example, rotation angles up to a few tens of degrees, shear factors bounded by a small absolute value, or contrast/brightness factors within a bounded interval. After these operations, we apply a cutout transform that removes a randomly located square patch of the image, with the side length sampled as a fraction of the image size. All sampling is done independently for each image and for each view, so that different augmented views of the same sample are stochastically decorrelated.

Augmentation pool for grayscale images. For grayscale datasets, we use a variant of the above policy tailored to single-channel images. In this case the transformation pool contains only operations that are meaningful on grayscale inputs:

- Geometric transforms: rotations, shears, and translations in the horizontal and vertical directions;
- Photometric transforms: inversion, histogram equalization, solarization, brightness, contrast, and sharpness adjustments;
- Cutout with a square mask whose side length is sampled up to a fixed proportion, filled with a constant gray or black intensity.

As in the color case, we randomly select a small number of transforms for each image, sample their magnitudes from predefined ranges, and apply them sequentially. The parameters of the grayscale policy such as the number of operations, the global magnitude index, and optional magnitude jitter are shared across the dataset, while the specific operations and their sampled magnitudes vary independently from image to image.

Usage in pretraining and supervised training. During contrastive pretraining, the above policies are used to create multiple stochastic views of each input example. All samples in a pretraining mini-batch participate in the SimCLR loss via their augmented views, while only the labeled samples in the batch participate in the SupCon loss. In the subsequent supervised training stage with GLL, we continue to apply the same augmentation pipeline to every training sample before feeding it through the encoder and the classifier. In this way, both the contrastive pretraining and the supervised training phases benefit from the same set of strong yet label-preserving data transformations.

2.6.2 Supplementary Adversarial Results

2.6.2.1 Visualization of Attacks

Adversarial attacks have to balance a trade-off between effectiveness (measured by decay in accuracy of the model) and detectability. Detectability is often quantified by a distance

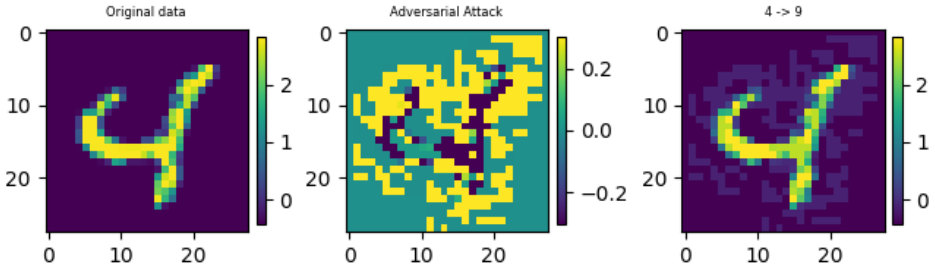
metric, but can be evaluated qualitatively - attacks that are successful but are not noticeable to the human eye are desireable, as this is the type of attack that poses the biggest threat in real-world applications. In Figure 2.10, we visualize natural and attacked images (along with their corresponding attacks) for FGSM, IFGSM, and CW attacks on MNIST of the naturally trained GLL model. Comparing the left and right-most columns, we see that FGSM and IFGSM models are easily detectable by the human eye, whereas the CW attacks are much more focused, and much less noticeable to the human eye. CW attacks are known to be much "smarter" than fast gradient methods (at the cost of more compute); GLL's overall robustness to all three attacks is a strong result for our proposed model.

2.6.2.2 Replacing an MLP Classifier with GLL in Testing

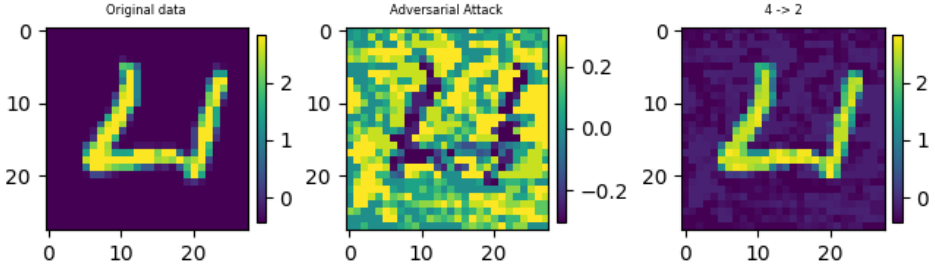
Recall the MNIST setting from Section 2.4.6.3, where we used a "small CNN". To further demonstrate the adversarial robustness of GLL, we replaced the final linear layer and softmax classifier used in the "small CNN" with GLL *during the attacks and testing* and *only during testing*. That is, we trained in a normal fashion, before replacing the softmax classifier with GLL for either the attacking (that is, the attacks use the gradient information from GLL) and inference phases, or just for the inference phase. The results are presented in Figure 2.11. We see that - surprisingly - when we use the gradients from GLL for the attacks, the model is much more robust to FGSM. This supports the idea that GLL is more difficult to attack than the usual linear layer and softmax classifier.

2.6.2.3 Further CW Experiments

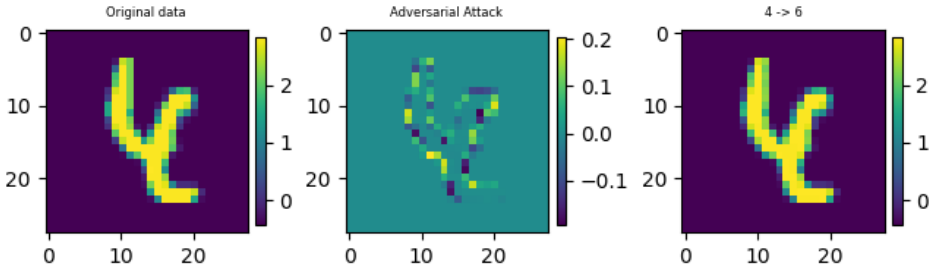
In Figure 2.9, we showed that for a fixed c, GLL resists the adversary from moving the images compared to the softmax classifier. However, we also wanted to demonstrate that across *distances* in the CW attacks, GLL is also superior. To do so, we ran further experiments where we increased the value of c so that the average distance increased further



(a) FGSM



(b) IFGSM



(c) CW

Figure 2.10: Visualization of FGSM, IFGSM, and CW attacks. The left, middle, and right columns show the original image, the attack, and the adversarial image, respectively. The title in the rightmost column indicates the original classification and the classification after the attack. These images were chosen to demonstrate attacks where the model was fooled by the adversary. [BCH24, Figure 10]

to the right. The results are shown in Figure 2.12. For MNIST, we ran additional experiments with $c = 10^4, 10^5, ..., 10^9$. For CIFAR-10, we ran additional experiments with $c = 2000, 5000, 10^4, 10^5, 10^6$.

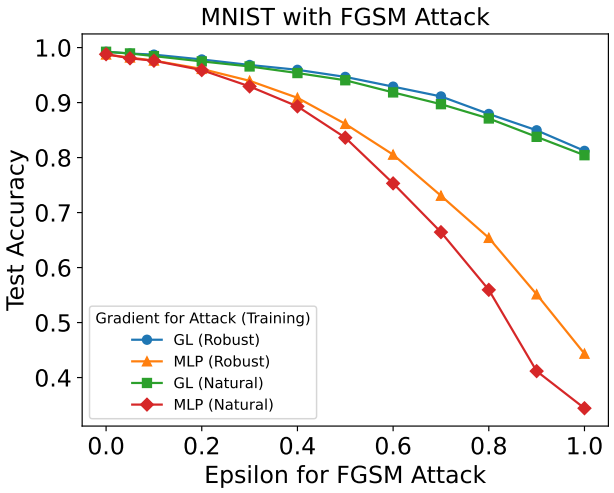
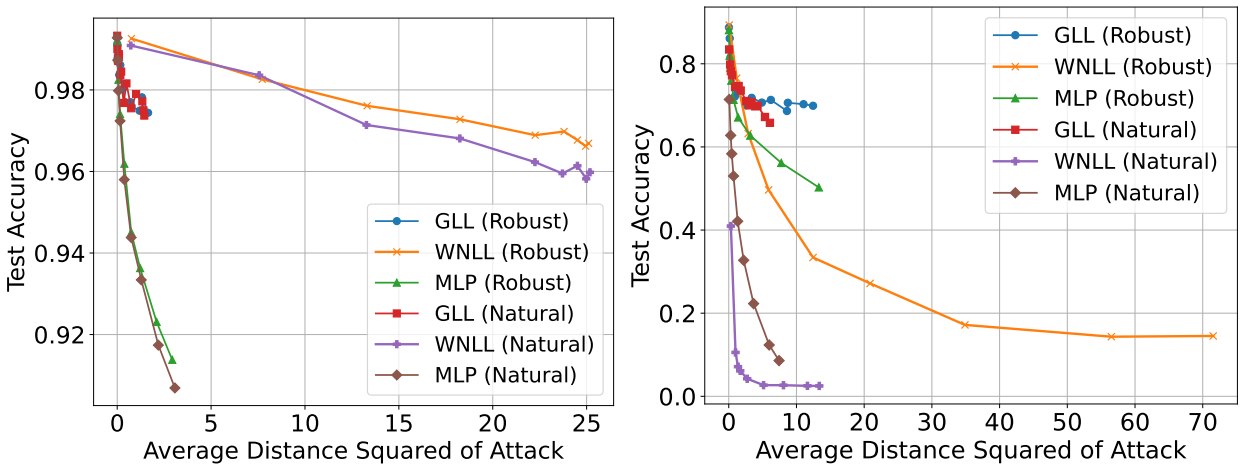


Figure 2.11: FGSM results on MNIST. The backbone network was trained in the standard way (with a softmax classifier). In the legend, MLP indicates that the FGSM attacks used the gradients from the MLP classifier (ie a linear layer and a softmax), whereas GLL used our graph learning layer. In all cases, the final classifier on the attacked images was GLL. Hence, the only difference between the results is which gradient was used for the attack. We see that the models that utilized the GLL gradients were significantly more robust to the FGSM attacks than using the standard MLP gradients. [BCH24, Figure 11]



(a) MNIST

(b) CIFAR-10

Figure 2.12: Comparison of accuracy vs. average distance of attack for CW attack on both datasets. Each point represents a different value of c, where we plot additional values of c for the GLL-based models compared to Figure 2.9. We see that GLL is still superior at resisting perturbations by CW attack, even when c is orders of magnitude larger than for WNLL or MLP, and even when controlling for distance on CIFAR-10. [BCH24, Figure 12]

CHAPTER 3

Topology-Aware Active Learning on Graphs

This Chapter is a version of [HMW25], currently under review at Pattern Recognition, which is a joint work with Jack Mauro, Adrien Weihs, and Andrea L. Bertozzi. I am the lead author. Jack, Adrien, and I proposed the initial ideas. I developed and implemented the curvature-based coreset and active learning methods. Jack and I implemented the hypergraph approaches and ran the experiments. Adrien wrote the initial hypergraph codebase and provided consistent feedback throughout. The three of us all wrote the paper. Andrea supervised the project and provided feedback on drafts of the paper.

While Chapter 2 was primarily concerned with learning a favorable graph structure, this Chapter focuses on how to leverage the graph structure for label-efficient learning. In this low label rate regime, GBSSL is often complemented by graph-based active learning (AL), which judiciously selects unlabeled points to query for labels to improve classifier performance. This chapter studies the *exploration-exploitation trade-off* in AL, where the learner must balance querying unexplored regions of the graph with exploiting existing knowledge to improve accuracy. Employing a notion of graph curvature, we design a coreset method that selects initial labeled points according to the graph's topology and detects when exploration is complete, improving on prior work that only considered path distances on the graph. We show that curvature can also replace rigid heuristics to improve the transition from exploration to exploitation in existing AL algorithms. Finally, we propose a variant of hypergraph learning that localizes multiscale interactions, speeding up runtimes by an order of magnitude while preserving higher-order regularization around labeled nodes that

strengthen exploitation. Experiments on image datasets showcase improved efficiency and accuracy at low label rates compared to existing benchmarks in coreset selection, sequential AL, and batch AL.

3.1 Introduction

GBSSL is often motivated by and used in applications where labeled training data is expensive to obtain, while unlabeled data is relatively easy to collect. A critical problem in these settings is which few points to select to label for training; given a small label budget, the choice of which points to label can drastically impact the underlying model's performance. To remedy this, AL iteratively builds the training set for a classifier based on an *acquisition function* that quantifies the "usefulness" of labeling each point in the unlabeled set, enabling careful selection of labeled points to maximize performance. Queries can be made sequentially (i.e., one at a time, updating the underlying classifier after each query) or in batches, depending on the application [Set09, CCT23]. When combined, GBSSL and AL can produce state-of-the-art results in a variety of applications, often rivaling or beating supervised neural networks while using significantly less training data [MMS22, CMB25, BOC23, CCT23].

A key challenge in AL is the *exploration–exploitation trade-off* [MC23]: an active learner may use its queries to label unrepresented regions of the graph (explore), or label along the classifier's existing decision boundaries to refine and improve predictions (exploit). Generally, exploration must precede exploitation; premature exploitation can lead to severe misclassifications by failing to sample entire regions of the input space. However, excessive exploration becomes label inefficient by failing to refine decision boundaries in later stages. Our approach addresses this trade-off in three novel ways:

- improves exploration by selecting representative and diverse initial labels using the graph's topology, whereas previous work only considered path distances;

- strengthens exploitation by proposing a novel hypergraph-based method that localizes higher-order regularization near informative nodes;
- provides a graph-driven signal that guides when to transition from exploration to exploitation, replacing hand-tuned heuristics in prior work.

To achieve strong exploratory coverage, we construct an effective *coreset* - an initial set of labeled nodes selected purely from the graph structure and without any reliance on the classifier. A well-chosen coreset should provide sufficient coverage and reflect the diversity of the data distribution. This ensures that once exploitation begins, the acquisition function has enough information to make an effective query. This is especially important when AL is done in batches [CCT23]; insufficient exploration can lead to catastrophic batch queries.

The main challenge of coreset selection is formalizing meaningful coverage of the graph. This problem was addressed in [CCT23]. The authors introduced Dijkstras' Annulus Coreset (DAC), which uses shortest-path distances to define metric balls on the graph, selecting nodes that collectively provide broad, quasi-uniform coverage. However, path distances do not account for the cluster structure of the graph (see Figure 3.1). To address this, we introduce a graph curvature-based approach to coreset selection which leverages community structure in the graph and improves coverage beyond simple shortest-path distances.

Once exploration yields a representative coreset, we shift focus to improving exploitation. To achieve this, we propose a novel approximation to multiscale Laplacian regularization, incrementally updating the graph Laplacian by incorporating higher-order structure only where it is most impactful: around labeled nodes. This localized rewiring strategy preserves the sparsity of the graph while harnessing the benefits of multiscale regularization, achieving a favorable trade-off between accuracy and efficiency. Experiments demonstrate this approach significantly outperforms standard Laplace learning, while running over an order of magnitude faster than full multiscale methods.

Our contributions: This chapter introduces a novel active learning framework lever-

aging *Balanced Forman Curvature* (BFC) [TGC22] and multiscale graph Laplacian regularization [MNW23, WBT25b, WBT25a] to improve both exploration and exploitation. Specifically:

1. **Curvature coreset selection**: We develop a novel greedy algorithm, called *Curvature Coreset* (CC, Algorithm 2) that constructs a coreset via a minimax formulation of a BFC-based objective, promoting topologically-aware, diverse coverage across communities of the graph. Our method is hyperparameter-free and features a natural, graph-theoretic stopping criterion to determine when exploration is complete.

2. **Principled exploration-exploitation balance via BFC**: By integrating our BFC-based criterion into Poisson ReWeighted Laplace Learning with τ-regularization (PWLL-τ) [MC23], we introduce a data-driven mechanism that adaptively transitions from exploration to exploitation, replacing heuristic, fixed-schedule baselines used in prior work.

3. **Localized Graph Rewiring for Label Propagation**: We enhance local graph structures around labeled nodes using a novel lightweight multiscale construction on a hypergraph, leading to significant performance gains at minimal computational cost.

4. **Strong empirical validation**: We demonstrate consistent gains over strong baselines across image benchmark datasets in both coreset quality, computational efficiency, and downstream AL performance. Our code is freely available at: `https://github.com/hardiman-mostow/TopologyActiveLearning`.

The remainder of the Chapter is structured as follows: in Section 3.2, we review the mathematical background relevant to our proposed methods; in Section 3.3, we present our graph topology-based framework for active learning; in Section 3.4, we evaluate its effectiveness through numerical experiments; and in Section 3.5 we conclude with a discussion of potential directions for future work.

3.2 Background and Notation

3.2.1 Graph Laplacian Regularization

This section presents the mathematical foundations for GBSSL broadly, as well as the specific tools that underpin our methodology. As in Chapter 2, we repeat some concepts from Chapter 1.2 for the sake of clarity and use self-contained notation conducive to clear exposition of our methods.

3.2.1.1 Single-scale Graph Construction

Given a dataset $\mathcal{X} := \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ embedded in a feature space, GBSSL leverages a weighted *similarity graph* that captures pairwise similarity between points. The similarity graph is represented using a weight matrix $W \in \mathbb{R}^{n\times n}$ with entries w_{ij}, where w_{ij} is larger the more similar x_i and x_j are. The weights are typically computed using non-increasing kernel function $\eta : [0,\infty) \to [0,\infty)$, a scale parameter $\varepsilon > 0$, and setting the edge weights as $w_{\varepsilon,ij} = \eta\left(\frac{\|x_i - x_j\|}{\varepsilon}\right)$. This kind of similarity graph is known as a ε-graph or a random geometric graph [Lux07]. A canonical choice of η is the indicator function $\eta(x) = \mathbb{1}_{\{[0,1]\}}(x)$, so that $w_{\varepsilon,ij} \neq 0$ if and only if $\|x_i - x_j\| \leq \varepsilon$. The scale parameter ε thus determines the radius of connectivity and plays a critical role in shaping the graph structure.

An alternative graph construction is a k-Nearest Neighbor (kNN) graph [Lux07]. While ε-graphs and kNN graphs are closely related, ε-graphs are often more convenient for theoretical analysis, whereas kNN graphs are typically preferred in practice [ST19, CCT20]. We include the subscript ε to specify ε-graph construction when necessary for theoretical precision, and omit it for simplicity when statements apply more broadly. Our experiments in Section 3.4 use kNN graphs.

To enable localized regularization in Section 3.3.3, we define a local analogue of the weight matrix. Let $\mathcal{S} \subset \{x_1, \ldots, x_n\}$ be a subset of selected points. We define the localized

weight matrix $W_\varepsilon^{\mathcal{S}} \in \mathbb{R}^{n\times n}$ as

$$(W_\varepsilon^{\mathcal{S}})_{ij} = \begin{cases} w_{\varepsilon,ij} & \text{if } x_i \in \mathcal{S} \text{ or } x_j \in \mathcal{S}, \\ 0 & \text{otherwise,} \end{cases}$$

i.e., we retain only edges adjacent to nodes in $\mathcal{S}$ and set all other entries to zero.

3.2.1.2 Standard Graph Laplacian Regularization

Many graph-based SSL methods are formulated as variational problems, where the goal is to find a labeling function u defined as:

$$u = \operatorname*{argmin}_{v:\{x_i\}_{i=1}^n \to \mathbb{R}^p} J(v) + \Psi(v, \ell), \tag{3.2.1}$$

where $J(v)$ is a regularization term promoting smoothness (according to the graph structure encoded in W), and $\Psi(v, \ell)$ enforces consistency with the known labels ℓ. A canonical example is *Laplace learning* [ZGL03] where the regularizer takes the form

$$J(v) = \sum_{i,j=1}^n w_{ij}(v(x_i) - v(x_j))^2, \tag{3.2.2}$$

promoting smoothness of the labeling function across connected nodes in the graph. Given the graph $G = (\mathcal{X}, W)$, the (unnormalized) *graph Laplacian* is defined as

$$L = D - W,$$

where D is the diagonal degree matrix with entries $D_{ii} = \sum_{j=1}^n w_{ij}$. Identifying a function $v : \{x_i\}_{i=1}^n \mapsto \mathbb{R}$ with a vector in $\mathbb{R}^n$, the regularizer in Laplace learning (3.2.2) can be rewritten as

$$J(v) = v^\top L v$$

which directly shows that L is symmetric and positive semi-definite [Lux07, Proposition 1]. We note that normalized variants of the graph Laplacian, such as the symmetric or

random-walk Laplacian, are also frequently used [Lux07]. Laplace learning uses hard label constraints

$$\Psi(v,\ell) = \begin{cases} 0 & \text{if } v = \ell \text{ on labeled nodes,} \\ +\infty & \text{otherwise.} \end{cases}$$

In the case of binary labels, $\ell(x) \in \{0,1\}$), final labels $\overline{\ell}_i$ are assigned by thresholding, i.e. $\overline{\ell}(x_i) = \mathbb{1}_{[0,0.5]}(u(x_i))$ for $i \in \mathcal{U}$. This formulation naturally extends to the multiclass setting with K classes. Let $S \in \mathbb{R}^{n\times K}$ be the one-hot label matrix (with rows $e_{\ell(x_i)}$ for $x_i \in \mathcal{L}$ where $e_j \in \mathbb{R}^K$ is the unit vector in j-th direction). Then, we use the regularizer

$$J(v) = \mathrm{Tr}(v^\top L v)$$

for $v \in \mathbb{R}^{n\times K}$ and label-fidelity term

$$\Psi(v,\ell) = \begin{cases} 0 & \text{if } v_{i,:} = S_{i,:} \text{ on labeled nodes,} \\ +\infty & \text{otherwise.} \end{cases}$$

Here $M_{i,:}$ denotes the i-th row of the matrix M. Once the optimal u is computed, predicted class labels are given by:

$$\overline{\ell}_i = \operatorname*{argmax}_k u_{ik}.$$

For notational simplicity, most of the remainder of the discussion will return to the binary classification case. When dealing with an ε-graph, $G = (\{x_i\}_{i=1}^n, W_\varepsilon)$ we write L_ε for its Laplacian. Using $W_\varepsilon^{\mathcal{S}}$, we also define the corresponding localized graph Laplacian $L_\varepsilon^{\mathcal{S}}$.

3.2.2 Active Learning on Graphs

In graph-based AL, the goal is to identify valuable points from the unlabeled set $\mathcal{U} = \mathcal{X} \setminus \mathcal{L}$ to add to the labeled set $\mathcal{L} \subset \mathcal{X}$ to improve the current labeling function $u \in \mathbb{R}^{n\times K}$. The initial labeled set $\mathcal{L}_0$ is called a *coreset* and subsequent querying is governed by an acquisition function $\mathcal{A} : \mathcal{X} \to \mathbb{R}$. At each iteration, the algorithm queries the labels of the point

$$x_{\mathrm{acq}} = \operatorname*{argmax}_{x\in\mathcal{U}} \mathcal{A}(x).$$

Algorithm 1 Active Learning (Sequential)

Input: Dataset $\mathcal{X}$, number of iterations k, acquisition function $\mathcal{A}(\cdot)$
Output: Optimized labeled set $\mathcal{L}_k$ and corresponding classifier u

1: Initialize coreset $\mathcal{L}_0 \subset \mathcal{X}$, set $\mathcal{U}_0 = \mathcal{X} \setminus \mathcal{L}_0$
2: **for** $i = 0$ to $k - 1$ **do**
3: Train classifier u_i using labeled set $\mathcal{L}_i$
4: Select query point:

$$x_{\text{acq}} = \underset{x \in \mathcal{U}_i}{\operatorname{argmax}}\, \mathcal{A}(x)$$

5: Update labeled and unlabeled sets:

$$\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{x_{\text{acq}}\}, \quad \mathcal{U}_{i+1} = \mathcal{U}_i \setminus \{x_{\text{acq}}\}$$

6: **end for**
7: Train final classifier u using labeled set $\mathcal{L}_k$

See Algorithm 1 for an overview of the AL loop. Among the most common choices for $\mathcal{A}$ is *uncertainty sampling*. This method quantifies uncertainty in the classification by computing the *margin*, or the gap between the predicted label $\overline{\ell}_i$ and the next most likely label:

$$\text{Margin}(x_i) = \overline{\ell}_i - \left(\underset{k \neq \overline{\ell}_i}{\operatorname{argmax}}\, u_{ik}\right)$$

To fit the convention of acquiring the point with the maximum acquisition value, the acquisition function is then

$$\mathcal{A}_{unc}(x_i) = 1 - \text{Margin}(x_i).$$

The choice of $\mathcal{A}$ can also be motivated by a Bayesian formulation [MB24], where the regularizer J defines a prior $\nu_1(v) \propto e^{-J(v)}$ and the fidelity term Ψ defines a likelihood $\nu_2(\ell \mid v) \propto e^{-\Psi(v,\ell)}$. The posterior is then given by:

$$\mu(v \mid \ell) \propto e^{-J(v)-\Psi(v,\ell)},$$

and the MAP estimate coincides with the minimizer of the original variational problem (3.2.1). This perspective naturally supports uncertainty quantification [BLS18], enabling

acquisition functions based on label variance [JH12, MGS13] or expected change in the underlying model [MB24].

Many AL methods rely on the current classifier u to guide label selection. However, when label information is scarce, this dependence can lead to degeneracies in the AL process, as the acquisition function cannot reliably distinguish which points are truly informative. For example, uncertainty sampling - an "exploitative" active learner that selects points near the current decision boundary of u - can perform poorly when the classifier is unaware of how many decision boundaries the dataset contains, over-refining some while ignoring others [MC23]. Alternatively, one can design AL acquisition functions that are primarily "explorative". Examples include V-Opt [JH12], Σ-Opt [MGS13], LAND [MM18], and CAL [CM21]. However, these methods can be much more expensive to compute [MC23]; for example, V-Opt and Σ-Opt require computing and storing an $n \times n$ covariance matrix after each new label acquisition. They also lack the ability to exploit once exploration is complete, limiting the empirical performance and label efficiency.

In contrast, coreset methods - such as DAC and the one proposed in this chapter - rely solely on the graph structure, which is sparse and only computed once, and have built-in stopping conditions that trigger once exploration is complete. Hence, to balance exploring and exploiting, one can use coreset selection to handle exploration before switching to unceratinty sampling-based exploitation. By first ensuring that the AL model has access to sufficiently informative labels via a coreset, we minimize the risk of such undesirable behaviors before the AL process begins, while maintaining computational and label efficiency.

An orthogonal approach to the "coreset and exploitative AL" paradigm is to design an acquisition function that can both explore and exploit. One such method is PWLL-τ [MC23], which we later describe in its own subsection. A key difference is that PWLL-τ assumes access to at least one label per class at initialization, while coreset-based algorithms make no such assumptions. However, as a further extension of our coreset methodology, we will show that the transition from exploring to exploiting in PWLL-τ can be optimized with our

same graph curvature-based approach.

Finally, thus far we have considered *sequential* active learning, where each iteration only queries one point. *Batch* active learning [CCT23], conversely, selects a query set $\mathcal{Q}$ of size B, $|\mathcal{Q}| =: B > 1$. This improves efficiency by requiring fewer computations of the acquisition function $\mathcal{A}$ (which also requires recomputing the underlying classifier). It also lowers the human labeling time by allowing several experts to work in parallel on a batch of queries, which can be a bottleneck in practice. However, batch AL requires care in designing a batch policy that avoids querying several points with redundant information; naively selecting the top B minimizers may result in several queries adjacent to one another. One such policy is LocalMax [CCT23], which requires query points in a batch to have a larger acquisition value than any of their neighbors, ensuring some diversity in queries. Batch AL also requires better coreset exploration than sequential AL; poor initial coverage can compound errors since many points are queried at once. Our experiments in Section 3.4 demonstrate that our proposed coreset method consistently outperforms existing approaches in both sequential and batch AL settings.

3.2.2.1 Dijkstra's Annulus Coreset

The *Dijkstra's Annulus Coreset* (DAC) algorithm [CCT23] constructs a coreset by iteratively selecting nodes from a graph $G = (\mathcal{X}, W)$ under two constraints:

- Separation: any two labeled nodes must be at least a graph distance r apart;
- Coverage: every node in the graph must lie within distance R of some labeled node.

Here, the graph distance d_G is defined by the shortest-path metric (via Dijkstra's algorithm).

Let $\mathcal{L}_0 \subset \{x_i\}_{i=1}^n$ denote the current set of labeled nodes and consider $r < R$. At each iteration, the algorithm computes:

- the seen set, i.e. all nodes within distance r of any labeled point:

$$\mathcal{S} = \bigcup_{x \in \mathcal{L}_0} B_r(x), \text{where } B_r(x) := \{y \in \{x_i\}_{i=1}^n : d_G(x, y) < r\},$$

- and the candidate set, i.e. all nodes within distance R but outside the seen set:

$$\mathcal{C} = \left(\bigcup_{x \in \mathcal{L}_0} B_R(x) \right) \setminus \mathcal{S}.$$

A new point x^* is selected uniformly at random from $\mathcal{C}$ and added to $\mathcal{L}_0$. If $\mathcal{C} = \emptyset$ but $\mathcal{S} \neq \{x_i\}_{i=1}^n$, the algorithm samples x^* uniformly from $\{x_i\}_{i=1}^n \setminus \mathcal{S}$ instead. After each selection, the sets are updated via:

$$\mathcal{L}_0 \leftarrow \mathcal{L}_0 \cup \{x^*\},\ \mathcal{S} \leftarrow \mathcal{S} \cup B_r(x^*),\ \mathcal{C} \leftarrow (\mathcal{C} \cup B_R(x^*)) \setminus B_r(x^*).$$

The process terminates once $\mathcal{S} = \{x_i\}_{i=1}^n$, ensuring complete coverage. The resulting set $\mathcal{L}_0$ serves as the DAC coreset.

In practice and for simplicity, r is set to be $R/2$. The performance of AL with DAC is sensitive to the choice of radii R, which are fixed, heuristically selected, and do not adapt to the underlying graph structure. Even for a fixed R, the size of the coreset can vary significantly due to the stochasticity, which limits applicability in settings where there is a fixed label budget or high cost of labeling (see 3.7.2). Moreover, DAC proceeds until the entire graph is covered, which may be unnecessary and inefficient in practice. In Section 3.3.1, we introduce a more adaptive and cluster-aware coreset selection method based on graph curvature, which improves both performance and label efficiency by leveraging the graph's intrinsic topology.

3.2.2.2 Poisson ReWeighted Laplace Learning with τ-regularization

An influential example of a jointly designed classifier-acquisition function pair is PWLL-τ [MC23]. PWLL-τ introduces an additional regularization term into the variational problem

(3.2.1), designed to enforce decay of the labeling function u away from labeled points:

$$u = \underset{v:\{x_i\}_{i=1}^n \to \mathbb{R}^d}{\operatorname{argmin}} \sum_{i,j=1}^{n} \gamma(x_i)\gamma(x_j) w_{ij} \|v(x_i) - v(x_j)\|^2 + \tau \sum_{i \in \mathcal{U}} \|v(x_i)\|^2$$
$$\text{subject to} \quad v(x) = e_{\ell(x)} \quad \text{for } x \in \mathcal{L}.$$

Here, the reweighting function γ is computed by solving the graph Poisson equation:

$$\sum_{j=1}^{n} w_{ij} \left(\gamma(x_i) - \gamma(x_j)\right) = \sum_{x_k \in \mathcal{L}} \delta_{ik} - \frac{1}{N}, \quad \text{for all } 1 \le i \le n,$$

where $\delta_{ik} = 1$ if $i = k$, and 0 otherwise. The corresponding acquisition function is given by $\mathcal{A}(x) = \|u(x)\|_2$, which, due to the τ term, tends to favor points far from the current labeled set - thus promoting exploration[1].

To gradually shift from exploration to exploitation, the authors introduce a geometric decay schedule for $\tau \to 0$ over multiple iterations. The schedule is defined by

$$\tau_{n+1} = \mu \tau_n, \quad \text{with} \quad \mu = \left(\frac{\varepsilon}{\tau_0}\right)^{\frac{1}{2K}},$$

where $\varepsilon = 10^{-9}$, and τ_n is set to zero after $2K$ iterations. The parameter K controls how quickly the method transitions from exploration to exploitation, and is set to the number of classes in the dataset. This decay strategy is somewhat rigid and ad hoc: while it ensures a transition from exploration to exploitation, it requires prior knowledge of the number of classes to set the parameter K. This makes the method dependent on an pre-selected hyperparameter, rather than label locations and graph structure, limiting its applicability to problems where class counts or distributions are unknown. Moreover, this value of K is not optimal in general; when to switch from exploring to exploiting will depend on the topology of the graph. In Section 3.3.2, we introduce an alternative exploration-exploitation strategy that is adaptive and data-driven - leveraging the graph topology via curvature - and demonstrate that this strategy is more flexible and performant across different datasets with variable classification structures.

[1]The convention adopted in that work is to acquire the *argmin*, not argmax, of $\mathcal{A}(x)$.

3.2.3 Graph Curvature

Ricci curvature is a fundamental object of study in differential geometry that measures whether local geodesics diverge (negative curvature), converge (positive), or stay parallel (zero). Analogues of curvature have been adapted to graphs [Oll10] and studied extensively for oversmoothing and oversquashing [TGC22] in GNNs, due to their ability to identify problematic *bottlenecks* in the graph. Graph curvature has also been used for pooling [FW24] and community detection [TLW25]. Given an edge $x_i \sim x_j$ (that is, $w_{ij} > 0$), curvature is positive when x_i and x_j have mutual neighbors (cliques or triangles), zero when the edge is in a grid-like structure (4-cycles or squares), and negative otherwise (tree-like structure). The notion of curvature we consider is *Balanced Forman Curvature* (BFC) [TGC22] for its conceptual and computational simplicity, and leave others for future lines of research. To define BFC, we first need a few helpful definitions, originally formulated in Topping et al. [TGC22]:

Definition 3.2.1 (Neighborhoods of $x_i \sim x_j$)**.**

We define the following terminology to describe the neighborhoods of nodes and edges:

1. *Let* $N_1(i) = \{x_j \mid w_{ij} > 0\}$ *be the set of neighbors of* x_i*.*
2. $\Delta(i,j) := N_1(i) \cap N_1(j)$ *is the set of mutual neighbors of* x_i *and* x_j*, i.e., nodes forming triangles based at* $x_i \sim x_j$*.*
3. $\square^i(i,j) := \{x_k \in N_1(i)) \setminus N_1(j),\, x_k \neq x_j \mid \exists x_w \in (N_1(k) \cap N_1(j)) \setminus N_1(i)\}$*. More simply,* $\square^i(i,j)$ *is the set of neighbors of* x_i *forming squares (4-cycles) traversing* $x_i \sim x_j$ *without diagonals inside.*
4. *Finally, we let* γ_{max} *be a correction factor counting the maximum number of 4-cycles based at* $x_i \sim x_j$ *which include a common node.*

We are now prepared to define BFC, which is the key ingredient in our graph-based coreset selection algorithm.

Definition 3.2.2 (Balanced Forman Curvature)**.** *For an edge $x_i \sim x_j$ in the edge set E of a graph G, and letting d_i be the degree of node x_i, we define the Balanced Forman Curvature as:*

$$Ric(i,j) := -2 + \frac{2}{d_i} + \frac{2}{d_j} + 2\frac{|\Delta(i,j)|}{max\{d_i, d_j\}} + \frac{|\Delta(i,j)|}{min\{d_i, d_j\}} + \frac{(\gamma_{max})^{-1}}{max\{d_i, d_j\}}(|\square^i(i,j)| + |\square^j(i,j)|)$$

Another interpretation of BFC - and a central idea motivating our method - is that $\mathrm{Ric}(i,j)$ is negative when $x_i \sim x_j$ forms a bridge between two communities. Implicit in this observation is the data points x_i and x_j *belong to two different communities.* This is the exact challenge of ensuring exploration in coreset selection: efficiently acquiring diverse points from all regions of the graph. In Section 3.3.1, we leverage this intuition behind BFC to design a coreset selection algorithm that ensures meaningful coverage of the graph. We further show that BFC can *measure* when exploration is complete, which can inform a stopping condition of our method (Section 3.3.1.2) and a more flexible, data-driven transition to exploitation in PWLL-τ (Section 3.3.2).

3.3 Methods

3.3.1 Curvature Coreset Selection

3.3.1.1 Algorithm Description

As previously discussed, highly negative curvature on an edge $x_i \sim x_j$ indicates a "bridge between clusters." In graph-based coreset selection, sampling from different clusters is key to building an effective and label-efficient training set. Hence, our algorithm iteratively builds the coreset by choosing nodes that exhibit high negative curvature with respect to

the existing coreset nodes. This allows the algorithm to account for the cluster structure - not just path distances - when determining the coreset.

Algorithm 2 Curvature Coreset (CC)

Input: Dataset $\mathcal{X}$, Adjacency matrix A, number of points to label n, optional reduction parameter r, optional stopping condition check `StopCond`.
Output: Labeled coreset $\mathcal{L}$ of length n.

Choose an $x \in \mathcal{X}$ uniformly at random.
Initialize $\mathcal{L} \leftarrow \{x\}$
Initialize unlabeled set $\mathcal{U} \leftarrow \mathcal{X} \setminus x$
if Using `StopCond` **then** ▷ See Section 3.3.1.2
    Initialize streaming curvature values $C \leftarrow \emptyset$
end if
if r is not **None then**
    Reduce candidate points $\mathcal{U} \leftarrow \text{topDegree}(\mathcal{U}, r)$ ▷ See Remark 3.3.3
end if
for $k = 1$ to n **do**
    Compute $\text{Ric}(i, j)$ for each $x_i \in \mathcal{U}$ and $x_j \in \mathcal{L}$ ▷ Def. 3.2.2
    Compute $\hat{x} = \text{argmin}_{x_i \in \mathcal{U}} \max_{x_j \in \mathcal{L}} \text{Ric}(i, j)$
    if Using `StopCond` **then**
        $c = \min_{x_i \in \mathcal{U}} \max_{x_j \in \mathcal{L}} \text{Ric}(i, j)$
        $C \leftarrow C \cup c$
        if `StopCond`(C) **is True then**
            $C \leftarrow C \cup \{\hat{x}\}$
            $\mathcal{U} \leftarrow \mathcal{U} \setminus \hat{x}$
            Terminate Algorithm
        end if
    end if
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{\hat{x}\}$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \hat{x}$
end for

Our Curvature-based Coreset selection algorithm (CC) is detailed in Algorithm 2. At each iteration, we add an unlabeled point to the coreset via a minimax formulation of BFC computed between the current coreset and candidate unlabeled points. The algorithm can either be terminated once a user-specified number of points are labeled or our proposed stopping condition (Section 3.3.1.2) is triggered.

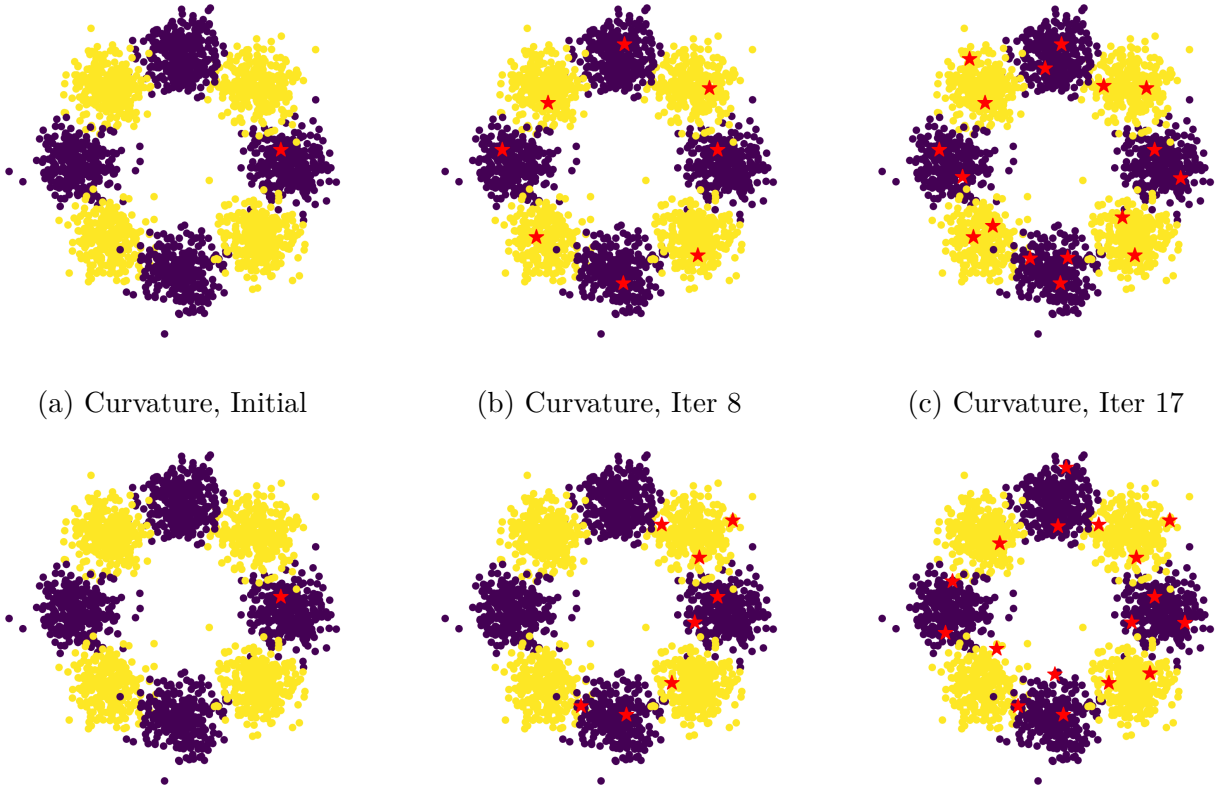
Remark 3.3.1 (Contrast to GNN Applications). Since we are interested in using curvature to

identify useful coreset *nodes* (compared to the GNN literature, which are typically concerned with bottleneck *edges*), we compute $\mathrm{Ric}(i, j)$ regardless of whether the edge $x_i \sim x_j$ exists. This ensures the search space at each iteration can include the entire set of nodes, not just those adjacent to current coreset nodes (which, typically, would be a poor choice to add to the coreset).

Remark 3.3.2 (Adjacency Matrix). Our algorithm uses only the binary adjacency matrix A of the graph (defined as $A_{ij} = 1$ if $W_{ij} > 0$, else $A_{ij} = 0$), not the weighted adjacency matrix W. While W contains more information than A, we found suitable notions of weighted curvature lacking in the current literature, particularly from a computational efficiency perspective. Moreover, using A was sufficient to achieve outstanding low-label rate classification results. Developing practical notions of curvature for weighted graphs is an interesting line of future research.

Remark 3.3.3 (Reduction Parameter). Inspecting Definition 3.2.2 and Algorithm 2 reveals a bias toward high-degree nodes. The $\frac{2}{d_i}$ and $\frac{2}{d_j}$ terms encourage the algorithm to choose points in high density regions, which tend to be points that are at the center - in some sense "representative" - of a cluster. This synergizes nicely with exploitative active learning, where queries tend to fall along the more sparsely-populated decision boundary. Moreover, we can use this bias to speed up the algorithm by only considering the top $1/r$ fraction of nodes *by degree* - reducing the search space on the graph by a factor of r - without sacrificing accuracy (we demonstrate this and discuss further in 3.7.1). This can be viewed as a method of *pruning* or *prefiltering*, which is common in the coreset literature [ZLL23].

We highlight and further motivate our method with an illustration on the "Blobs" dataset in Figure 3.1. This dataset (previously used in [MC23]) consists of eight Gaussian clusters with 300 points per cluster, centered at even spacings around the unit circle, with $\sigma = 0.17$, where the blobs are assigned alternating classes. We compare DAC with our proposed coreset algorithm, CC. DAC is slow to explore the dataset, requiring 17 iterations before each of

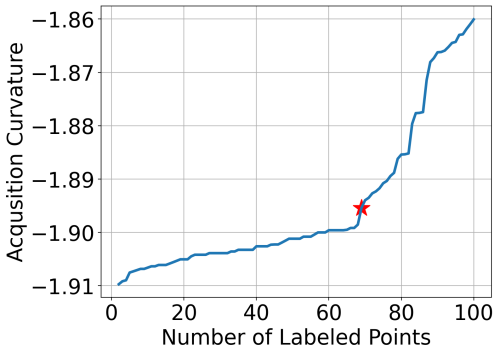


(a) Curvature, Initial (b) Curvature, Iter 8 (c) Curvature, Iter 17

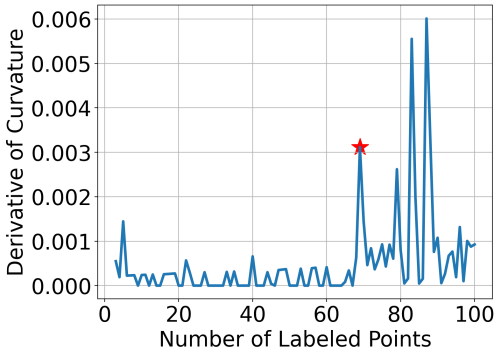
(d) DAC, Initial (e) DAC, Iter 8 (f) DAC, Iter 17

Figure 3.1: Coreset points chosen by CC and DAC at different iterations of coreset selection on the Blobs dataset. CC selects exactly one point from each of the eight clusters by the eighth iteration - the most efficient exploration of the cluster structure of the dataset possible. Moreover, they are all toward the center of each cluster (not outliers) due to the d_i terms in $\mathrm{Ric}(i, j)$. Conversely, DAC is inefficient, needing over twice as many iterations to sample from every cluster, and often sampling near the edge of the dataset. [HMW25, Figure 1]

the eight clusters has a point in the coreset. Meanwhile, CC samples from all eight clusters by iteration eight - over twice as fast as DAC. The curvature method is also biased toward higher-degree nodes, selecting points at the center of clusters, while DAC is random.



(a) Value of c (Line 15, Algorithm 2)



(b) First-order difference of the sequence of c's.

Figure 3.2: Illustration of our proposed stopping condition on MNIST. Across datasets, the value of c in Algorithm 2 steadily grows until a certain "saturation" point when it starts rapidly increasing. This indicates - according to the curvature metric - that the graph topology has been sufficiently explored, and exploitative active learning may begin. The red star indicates when the online Z-score stopping condition triggers (Algorithm 3), corresponding to the first large jump in curvature values among coreset points. [HMW25, Figure 2]

3.3.1.2 Stopping Condition

Depending on the application, a stopping condition may be preferable to a fixed label budget for the coreset. For example, DAC stops when every node on the graph is within R of a coreset point. This is often label-inefficient. Conversely, our stopping condition is online and data-driven, stopping when the underlying metric reflects that the graph is sufficiently explored.

Due to the minimax formulation of our coreset method, the value of curvature of the new coreset point between one iteration and the next will be nondecreasing. Figure 3.2 illustrates that after sufficient exploration, there is a relatively large, sudden increase in the value of curvature between the acquired point and the existing coreset. This points toward a simple stopping condition: we stop once a large enough "jump" is made in BFC. To quantify this,

Algorithm 3 Online Z-Score Stopping Condition

Input: Online sequence C of curvature values (see Line 15, Algorithm 2), Window size N, threshold z_{thresh}.
Output: Whether to end the coreset, **True** or **False**.

Initialize empty list $\mathcal{H} \leftarrow \emptyset$
for each new data point $c_t \in C$ **do**
    Append c_t to $\mathcal{H}$
    if $|\mathcal{H}| > N$ **then**
        Remove oldest element from $\mathcal{H}$
    end if
    if $|\mathcal{H}| < N$ **then**
        continue ▷ Not enough data yet
    end if
    Compute mean $\mu \leftarrow \text{mean}(\mathcal{H})$
    Compute std $\sigma \leftarrow \text{std}(\mathcal{H})$
    Compute $z \leftarrow |c_t - \mu|/\sigma$
    if $z > z_{\text{thresh}}$ **then**
        Return True and end the coreset
    end if
end for
Return False

we use a simple online anomaly detection algorithm based on rolling Z-scores. We record the discrete difference of successive values of BFC and compute a rolling mean and standard deviation of the last N (we use N=20) differences. For each new sample (difference), we compute the Z-score, and the stopping condition triggers if the Z-score exceeds a threshold (we use 3 standard deviations in our experiments). We provide pseudocode for this anomaly detector in Algorithm 3.

3.3.2 Principled Exploration–Exploitation Balance via BFC

As previously discussed, PWLL-τ has a set decay schedule for τ which controls the tradeoff between exploration and exploitation. We instead propose a data-driven BFC metric to decide when to *switch* from exploration to exploitation. We assume a fixed $\tau = \tau_0$ to begin and run PWLL as usual, without decaying τ_0. At each iteration, PWLL acquires a new

Algorithm 4 PWLL-τ with Curvature-based τ Schedule

Input: PWLL-τ classifier with initial $\tau = \tau_0 > 0$, minimum norm acquisition function $\mathcal{A}$, k from k-NN search, number of AL iterations n.
Output: Labeled set of points $\mathcal{L}$.

```
Initialize L = ∅
for k = 1 to n do
    Acquire new point x_i with A
    Compute Ric(i, j) for each x_j ∈ L                    ▷ Definition 3.2.2
    if max_{j∈L} Ric(i, j) > (−2 + 4/k) then              ▷ Equation 3.3.2
        Update τ = 0
    end if
    Update L = L ∪ {x_i}
end for
```

point according to the minimum norm acquisition function. Similar to the coreset method, we compute BFC between the new acquisition and the current set of all labeled points. This yields a data-driven metric indicating how explored the dataset is.

When the metric is close to -2, this indicates the current acquisition is not "close" to any other points (as measured by BFC, see Definition 3.2.2), and PWLL should continue exploring. However, once exploration is complete, the acquisitions will get closer to one another, and BFC will increase. Assuming a kNN graph, the following argument provides a quantitative bound on when acquisitions are still far enough apart to continue exploring:

$$\begin{aligned}
\mathrm{Ric}(i,j) &= -2 + \frac{2}{d_i} + \frac{2}{d_j} + 2\frac{|\Delta(i,j)|}{\max\{d_i,d_j\}} + \frac{|\Delta(i,j)|}{\min\{d_i,d_j\}} + \frac{(\gamma_{\max})^{-1}}{\max\{d_i,d_j\}}(|\square^i(i,j)| + |\square^j(i,j)|) \\
&= -2 + \frac{2}{d_i} + \frac{2}{d_j} && (3.3.1) \\
&\leq -2 + \frac{4}{k}, && (3.3.2)
\end{aligned}$$

where (3.3.1) is because we assume $\Delta(i,j) = \square^i(i,j) = \square^j(i,j) = \emptyset$ when exploration is still in progress and (3.3.2) is due to the fact that $d_i \geq k$ for all i in a kNN graph. Hence, an upper bound of $\mathrm{Ric}(i,j)$ *when* x_i *and* x_j *are not in the same neighborhood* (see Definition 3.2.1) is $-2 + \frac{4}{k}$. $\mathrm{Ric}(i,j)$ exceeding this amount indicates that x_i and x_j are in the same neighborhood, and thus exploration is complete. We describe our methodology in Algorithm

4.

3.3.3 Localized Graph Rewiring for Label Propagation

While effective in some settings, single-scale graph constructions are sensitive to the choice of scale parameter and may fail to capture both local and global structure in heterogeneous data. In *multiscale graph construction*, edge formation aggregates information across multiple neighborhood scales, enhancing representations of distinct clusters and density variations. The multiscale structure can be used to more effectively regularize within well-connected regions of the graph - such as dense clusters - by penalizing higher-order variations of the labeling function. For example, in multiscale graph Laplacian regularization [MNW23] and higher-order hypergraph learning [WBT25b, WBT25a], powers of the Laplacian are applied locally depending on the graph's structure, enabling adaptive smoothness control across different regions of the data.

Formally, we pick scale parameters $\varepsilon_1 > \cdots > \varepsilon_q$ and construct the associated Laplacian matrices $\{L_{\varepsilon_k}\}_{k=1}^q$. Picking powers $p_1 \leq \cdots \leq p_q$, we then use the regularizer

$$J(v) = \sum_{k=1}^{q} \lambda_k v^\top L_{\varepsilon_k}^{p_k} v = v^\top \left[\sum_{k=1}^{q} \lambda_k L_{\varepsilon_k}^{p_k} \right] v =: v^\top L^{(q)} v \tag{3.3.3}$$

where λ_k are fixed positive weights.

As k increases, the associated scale ε_k decreases, so the constructed graph retains only the strongest (most local) connections - emphasizing finer structural features and high-density neighborhoods. Under the standard graph homophily assumption [MLS22], these regions benefit from stronger regularization - encouraging smooth label variation across densely connected neighborhoods. This motivates using larger powers p_k for the corresponding Laplacians L_{ε_k}. See Figure 3.3 for a visual illustration of this intuition (e.g., with $p_k = k$).

The use of the multiscale regularizer $J(v) = v^\top L^{(q)} v$ in semi-supervised learning has been empirically validated in [MNW23]. Its theoretical properties - including well-posedness, extensions to supervised learning, connections to hypergraph models, and applications beyond

geometrically embedded data - have been further explored in [WBT25b, WBT25a]. Notably, it was shown that $L^{(q)}$ can itself be interpreted as the Laplacian of a derived graph, and its spectral characteristics have been analyzed in detail.

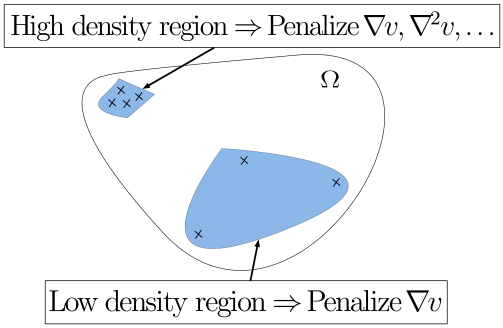


Figure 3.3: Higher-order smoothness is imposed on the labeling function v in denser regions, while allowing greater flexibility in sparser areas. [WBT25a, Figure 3].

Graph-based active learning can be greatly improved by incorporating multiscale Laplacian regularization, i.e., replacing the classical graph Laplacian with $L^{(q)}$, as demonstrated in [WBT25b]. However, computing $L^{(q)}$ incurs substantial computational cost: while the base Laplacian L_ε is typically sparse, its powers L_ε^k (for $k \geq 2$) grow increasingly dense.

Specifically, the entry (i, j) of L_ε^k is recursively given by $(L_\varepsilon^k)_{ij} = \sum_{r=1}^n (L_\varepsilon^{k-1})_{ir} \cdot (L_\varepsilon)_{rj}$, which expands explicitly to

$$(L_\varepsilon^k)_{ij} = \sum_{i_1, i_2, \ldots, i_{k-1}=1}^{n} (L_\varepsilon)_{ii_1} (L_\varepsilon)_{i_1 i_2} \cdots (L_\varepsilon)_{i_{k-1} j}.$$

By construction, $(L_\varepsilon)_{ij} \neq 0$ if and only if $w_{\varepsilon,ij} \neq 0$, i.e., there exists an edge between nodes x_i and x_j. Consequently, $(L_\varepsilon^k)_{ij} \neq 0$ if and only if there exists at least one path of length k between x_i and x_j. As k increases, the number of such paths - and thus nonzero entries - grows rapidly. This densification breaks the original sparsity pattern, resulting in significantly higher memory and computational costs.

To circumvent this issue, we propose a localized approximation to multiscale regularization by rewiring the graph at only the labeled nodes, where higher-order smoothing is most

beneficial. As discussed in Section 3.2.1.2, reinforcing regularization of the labeling function near given points promotes propagation of their labels to neighboring nodes. Since the acquisition function targets high-utility samples, concentrating regularization around them enhances exploitation - particularly in homophilous graphs, where nearby nodes often share labels. Our approach is detailed in Algorithm 5.

We refer to our method as rewiring: at each iteration, the matrix L in the regularizer $J(v)$ represents the Laplacian of an incrementally modified graph [WBT25b]. Crucially, the updates to L are additive and localized, enabling efficient incorporation of higher-order structure while avoiding the computation and densification of full Laplacian powers.

This approach hinges on the assumption that labeled nodes are particularly suitable targets for enhanced regularization, as they are *purposefully* selected by the acquisition function $\mathcal{A}(x)$. To test this hypothesis, we also compare our method to a control strategy that applies the same localized regularization, but around randomly selected nodes at each step, rather than those chosen by $\mathcal{A}(x)$ (see Section 3.4.5).

Algorithm 5 Localized Graph Rewiring for Label Propagation

Input: Dataset $\mathcal{X}$, labeling budget B, acquisition function $\mathcal{A}$, scale parameters $\varepsilon_1 > \cdots > \varepsilon_q$, powers $p_1 \leq \cdots \leq p_q$, positive coefficients $\{\lambda_k\}_{k=1}^q$.

Output: Labeled set $\mathcal{L}_B$ and corresponding classifier u

Compute weighted graph $G = (\{x_i\}_{i=1}^n, W_{\varepsilon_1})$ and matrix $L = L_{\varepsilon_1}^{p_1}$
Initialize coreset $\mathcal{L}_0 \subset \mathcal{X}$, set $\mathcal{U}_0 = \mathcal{X} \setminus \mathcal{L}_0$
for $i = 0$ to $B - 1$ **do**
    Train classifier u_i using labeled set $\mathcal{L}_i$ through Laplace learning ▷ Section 3.2.1.2
    Select query point: $x_{\text{acq}} = \text{argmax}_{x \in \mathcal{U}_i} \mathcal{A}(x)$
    Update labeled and unlabeled sets: $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{x_{\text{acq}}\}, \quad \mathcal{U}_{i+1} = \mathcal{U}_i \setminus \{x_{\text{acq}}\}$
    Update Laplacian:

$$L \leftarrow L + \sum_{k=2}^{q} \lambda_k (L_{\varepsilon_k}^{\{x_{\text{acq}}\}})^{p_k}$$

end for
Train final classifier u using labeled set $\mathcal{L}_B$

3.4 Numerical Experiments

3.4.1 Graph Construction and Active Learning Setup

For all experiments, following the discussion in Section 3.2.1.1, we construct kNN graphs[2], replacing the scale parameters ε_k with a decreasing sequence of neighborhood sizes $k_1 > \cdots > k_q$ ($q = 1$ except for the multiscale rewiring experiments in Section 3.4.5). This ensures that our weight matrices are sparse, lowering memory and compute overhead. Because GBSSL is a transductive paradigm, we embed and build the graph over the entire dataset, irrespective of built-in training-testing splits.

Edges are weighted with a gaussian kernel

$$w_{ij} = \exp\Big(\frac{-4\|x_i - x_j\|^2}{d_k(x_i)^2}\Big),$$

where $d_k(x_i)$ is the distance from x_i to its kth nearest neighbor, which ensures edge weights adapt to both sparse and dense regions of the feature space. The norm $\|\cdot\|$ is cosine similarity (angular metric). Importantly, weights are computed only for the k nearest neighbors identified via the kNN search; all other entries in W are set to zero.

Sections 3.4.2 and 3.4.3 present coreset and AL results on image datasets, comparing our proposed CC with other coreset methods on downstream sequential and batch AL, respectively. In Section 3.4.4, we report results on image datasets for our BFC-based metric that provides PWLL-τ a data-driven signal to inform the transition from exploration to exploitation. Finally, in Section 3.4.5, we adopt the localized rewiring strategy described in Algorithm 5. Across all experiments - except PWLL-τ in Section 3.4.4 - we use uncertainty sampling as the acquisition function (see Section 3.2.2). Throughout, plots display mean accuracy (solid line) with one standard deviation (shaded region) across trials.

[2]We use Annoy for approximate nearest neighbor searches `https://github.com/spotify/annoy`.

3.4.2 Coreset and Sequential Active Learning on Image Benchmarks

We present results on the MNIST [Yan10], FashionMNIST [XRV17], and CIFAR-10 [KH09] datasets. Before constructing the graphs, we first embed the data into a latent space using unsupervised neural networks, which provides a better setting to compute similarities and ensure a meaningful graph [BCH24, MMS22]. For MNIST and FashionMNIST, we utilize VAE [KW13] embeddings, and similarly we use SimCLR [CKN20] embeddings of the CIFAR-10 data, all provided by the `graphlearning` Python package [Cal22]. We construct the graph with $k_1 = 25$, and use Laplace learning as the classifier (see Section 3.2.1.2). We run 10 trials of each experiment. Our main results are shown in Figures 3.4 and 3.5, and Table 3.1. We discuss them further here.

Remark 3.4.1 (Presenting a Fair Comparison to DAC). DAC requires the user to specify a radius R that defines the distance from the current coreset points to the set of candidates for the next coreset point. The algorithm terminates once every coreset point is within at least R of every other point. Hence, the choice of R has a large impact on the number of points in the coreset, and thus also the accuracy. This leads to two issues: (1) since the algorithm is stochastic - randomly choosing a point from the set of candidate points at each iteration - it is impossible to ensure that the coreset will finish with a certain number of points, and (2) manually ending DAC early before it has reached its stopping condition could lead to catastrophic results; one could terminate DAC before a large portion of the dataset has been "seen" according to the choice of R.

Remark 3.4.1 highlights strengths of our CC method: it does not require any user-inputted parameter such as R - which has a significant impact on the size of the coreset and its accuracy - and greedily chooses the best point at each iteration. However, this presents some difficulty in reporting completely fair comparisons between our method and DAC at a given label rate, such as 100 labels in Figure 3.4. As we have established, issue (1) may cause DAC to end with fewer than 100 points, and truncating a coreset that exceeds 100

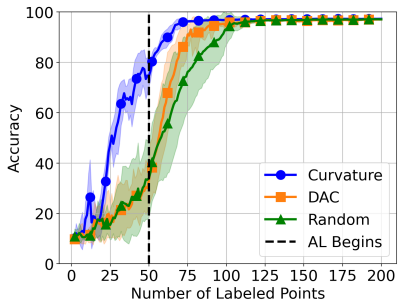
Method	MNIST		FashionMNIST		CIFAR-10	
	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
CC	94.7	39.6	64.1	38.1	79.7	22.1
DAC	68.7	62.1	48.9	76.1	74.2	46.2
Random	66.3	0.0	50.5	0.0	69.1	0.0

Table 3.1: Comparison of accuracy and efficiency for coreset methods after 100 labels, averaged over 10 trials. CC is consistently faster and performs better than DAC and Random. 100 labels is a 0.14% label rate for MNIST and FashionMNIST, and a 0.16% label rate for CIFAR-10. [HMW25, Table 1]

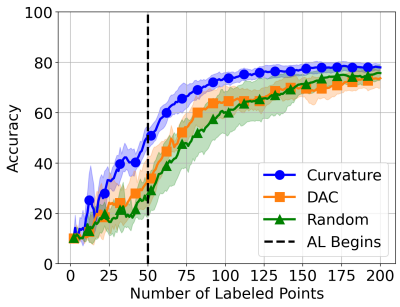
labeled points could significantly harm results per issue (2).

To present a fair comparison, our experiments with fixed-size coresets (Figure 3.4) use the following procedure: we fix a coreset label budget (50 or 100), and report DAC results where we pick an R that leads to approximately that many points. When the coreset ends, we start active learning (uncertainty sampling) and report results up to 200 total labels. For CC and Random, we always stop the coreset at 50 (Figure 3.4, top) or 100 (Figure 3.4, bottom) labeled points, since these do not have the same behavior as DAC. For DAC, we use $R = 0.25$, 0.4, and .35 for MNIST, FashionMNIST, and CIFAR-10, respectively to acquire approximately 50 points, and half those values for 100 points. 3.7.2 lists statistics on the coreset sizes obtained by DAC across each of the experiments, illustrating its high variability. For CC, we use a reduction parameter of $r = 100$ for MNIST and FashionMNIST and $r = 50$ for CIFAR-10.

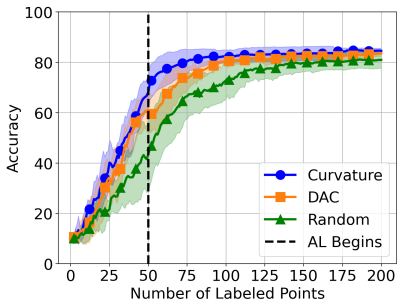
Figure 3.4 shows that our novel CC method significantly outperforms DAC and Random across all three datasets, and leads to faster convergence on downstream AL. For example, our method combined with AL achieves 80% accuracy on the FashionMNIST dataset with only 200 labels, corresponding to a 0.2% label rate. This is an over 10% absolute improve-



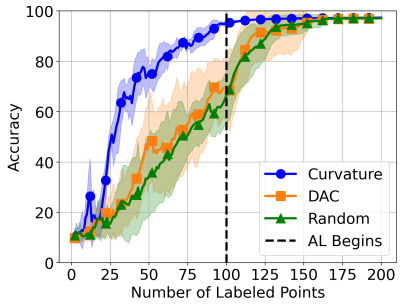
(a) MNIST, AL begins at 50 labels.



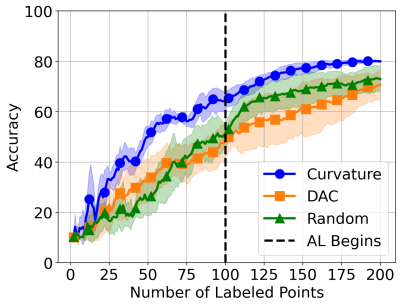
(b) FashionMNIST, AL begins at 50 labels.



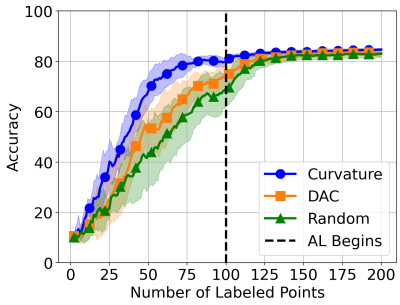
(c) CIFAR-10, AL begins at 50 labels.



(d) MNIST, AL begins at 100 labels.



(e) FashionMNIST, AL begins at 100 labels.



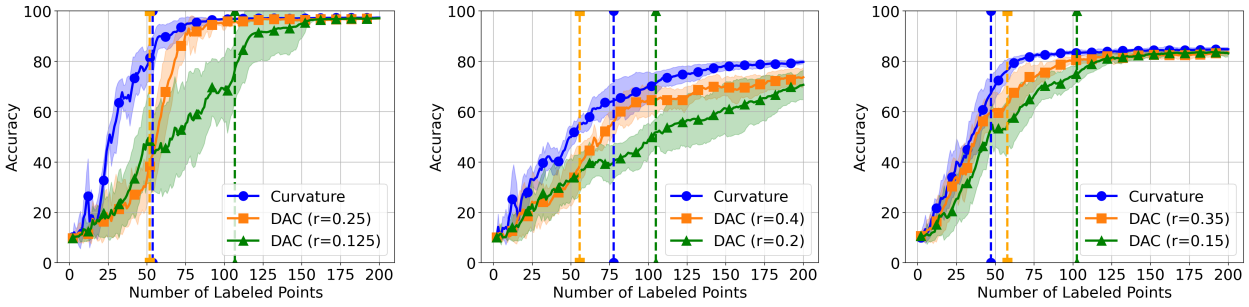
(f) CIFAR-10, AL begins at 100 labels.

Figure 3.4: Coreset and AL results for Curvature, DAC, and Random on several benchmarks. The top and bottom rows present results when AL begins at 50 and 100 labels, respectively. The solid line indicates the mean and shaded region indicates one standard deviation over 10 trials. Our method significantly outperforms the others, especially at the lower label rates; for reference, 50 labeled points is approximately 0.07% of the MNIST and FashionMNIST datasets and 0.08% of CIFAR-10. [HMW25, Figure 4]

ment over the other methods. On the other datasets, we see convergence of all three methods to approximately the same accuracy with AL, but CC does so with significantly fewer iterations (labels). We summarize our coreset results in Table 3.1, which includes timing details demonstrating that CC is faster to run than DAC, in addition to being more accurate.

Since the stopping condition of DAC is the crux of the issue outlined in Remark 3.4.1, we also present results where AL starts *after the coreset method's stopping condition is*

reached in Figure 3.5. Although this means AL may start at different iterations for different methods, it allows for a clear comparison of both methods (and their stopping conditions). Specifically, Figure 3.5 compares CC with DAC (at different values of R) where the coreset method ends according to its own stopping condition. A good stopping condition should terminate (and hence switch to AL) early enough to be label-efficient but not so early as to leave a portion of the graph unexplored. Figure 3.5 shows that - regardless of whether our method's stopping condition triggers before, during, or after DAC's - our method is more accurate before and during AL across datasets.



(a) MNIST (b) FashionMNIST (c) CIFAR-10

Figure 3.5: Coreset and AL accuracy comparison between our method and DAC (with different radii), where we use each method's stopping condition. The like-colored dashed line indicates where the stopping condition is triggered (and AL begins) for each method. Across datasets, and regardless of when stopping conditions are triggered, our method significantly outperforms DAC, particularly at lower label rates. [HMW25, Figure 5]

3.4.3 Sequential vs. Batch Active Learning

Another empirical criterion to evaluate a coreset method is to compare the performance of downstream sequential and batch AL with the same initial coreset. Batch AL offers significant efficiency savings at the cost of accuracy compared to sequential AL, but a carefully chosen coreset can minimize the difference [CCT23]. In a sense, exploitative batch AL requires even stronger coreset exploration than in the sequential case; the underlying classifier

must have sufficient information to determine not only one useful point to query, but several.

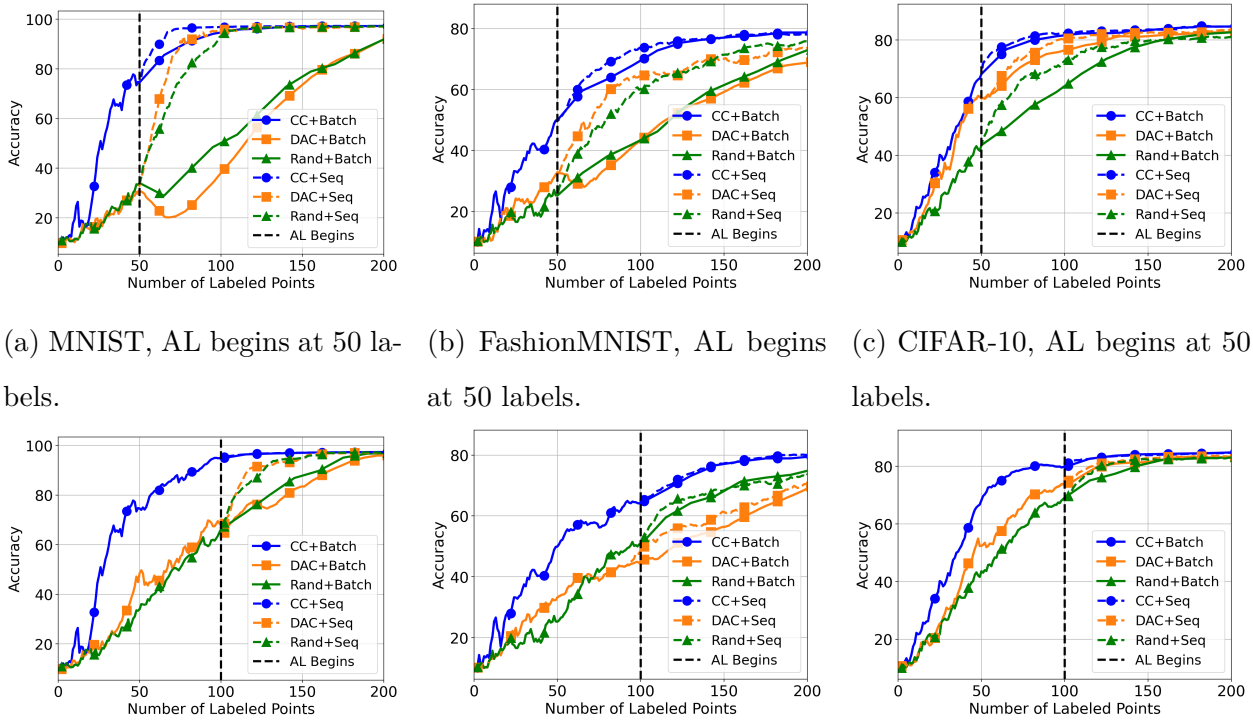
Figure 3.6 compares CC, DAC, and Random coresets on MNIST, FashionMNIST, and CIFAR-10, reporting both sequential and batch AL accuracy with Laplace learning over 10 trials[3]. We use LocalMax as our batch policy [CCT23] with a batch size of 15. Not only does CC provide the best coreset for both AL policies, but the decrease in accuracy between sequential and batch AL is minimal. Conversely, batch AL after DAC and Random is significantly worse than sequential AL - in some cases *decreasing* in accuracy when batch AL begins. This implies that these coreset methods are not selecting effective initial labels for querying in batches; there is enough information in the coreset to choose good sequential queries (and update the classifier more frequently), but not enough to choose good batches. Indeed, AL behavior with DAC coresets is much closer to Random than it is to CC, implying a topologically-aware, greedy strategy is better for ensuring meaningful exploration.

3.4.4 Application to PWLL

This section presents results on extending CC to provide a curvature-based signal for determining when PWLL-τ should transition from exploration to exploitation - that is, when to set $\tau = 0$. As discussed in Section 3.2.2.2, given a user-defined parameter K, standard PWLL-τ decays τ to zero over 2K acquisition steps. After that, the algorithm becomes fully exploitative.

We report results on 4 datasets: EMNIST [CAT17] (an extension of MNIST using letters, with 47 classes), FashionMNIST, CIFAR-10, and Box, a two-class toy dataset consisting of a 65×65 lattice of points on the unit square with a class boundary at $x = 0.3$, also used in [MC23]. We adopt the re-labeling scheme from [MC23] for EMNIST, FashionMNIST, and CIFAR-10. Specifically, we map original labels to their values modulo 5 for EMNIST

[3]Note that the sequential AL results in Figure 3.6 are the same as in Figure 3.4, but we reproduce them for a clear comparison to batch AL.



(a) MNIST, AL begins at 50 labels.

(b) FashionMNIST, AL begins at 50 labels.

(c) CIFAR-10, AL begins at 50 labels.

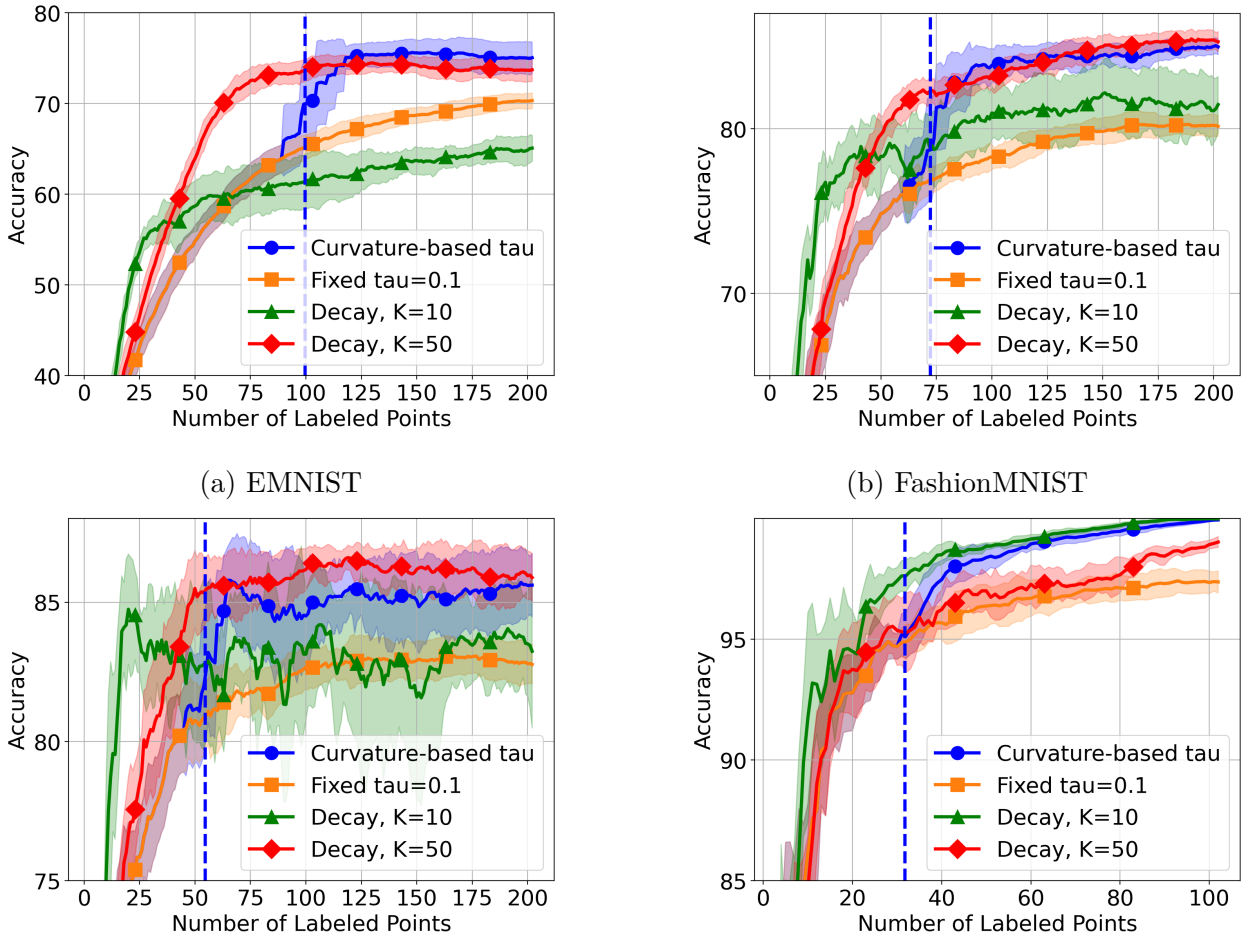
(d) MNIST, AL begins at 100 labels.

(e) FashionMNIST, AL begins at 100 labels.

(f) CIFAR-10, AL begins at 100 labels.

Figure 3.6: Coreset and AL results for Curvature, DAC, and Random on several image benchmarks. The top and bottom rows present results when AL begins at 50 and 100 labels, respectively. We compare downstream AL performance under batch (solid line) and sequential (dashed line) uncertainty sampling AL. We report the mean over 10 trials. We exclude standard deviation shading for clarity. Not only does CC significantly outperforms the others with both AL policies, but the performance drop between sequential and batch is much less severe. This implies that CC provides a much more meaningful coverage of the graph for batch AL than DAC or Random. [HMW25, Figure 6]

and modulo 3 for FashionMNIST and CIFAR-10, creating "modulo" classes (e.g., (0,3,6,9), (1,4,7), and (2,5,8) for FashionMNIST and CIFAR-10). Each experiment is initialized with one labeled example per "modulo" class (e.g., 3 total labels for FashionMNIST and CIFAR-10). This yields more complex class boundaries and ensures some original classes are initially



(a) EMNIST

(b) FashionMNIST

(c) CIFAR-10

(d) Box

Figure 3.7: Performance of PWLL-τ with different decay schedules. Using curvature to inform the value of τ consistently ensures effective transitioning from exploration to exploitation, and removes the need to set K beforehand in a decay schedule. The dashed line indicates the point at which our curvature-based method sets $\tau = 0$. [HMW25, Figure 7]

unlabeled, *requiring* the active learner to discover them through exploration. We construct the graphs with $k_1 = 25$, and run 10 trials. In all experiments, we initialize with $\tau_0 = 0.1$ and compare:

- Fixed τ throughout,
- Decay schedule from [MC23] for various K,

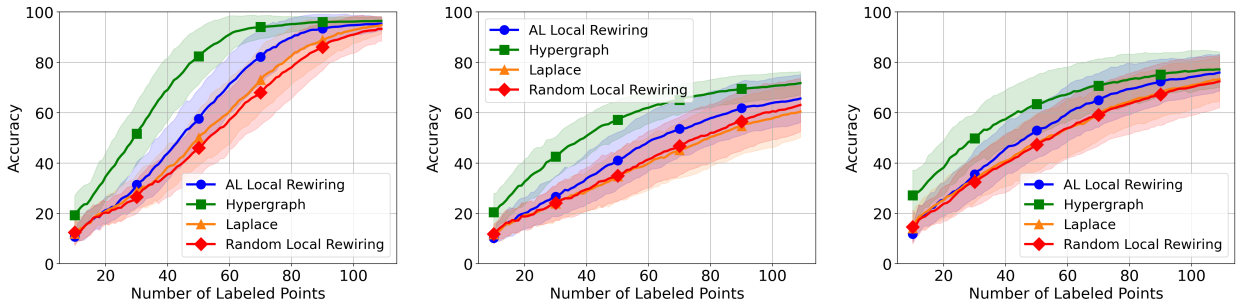
- Our curvature-based method for updating $\tau_0 = 0.1$ to $\tau = 0$.

Figure 3.7 shows that the optimal moment to switch from exploration to exploitation varies with the dataset and its classification structure, and that our data-driven signal reliably identifies this point without requiring any preset decay schedule. This is clearly illustrated by comparing the Box and EMNIST results.

On Box - a dataset with simple topology and classification structure - $K = 10$ outperforms $K = 50$ because the dataset requires less exploration before exploitation. Our method mirrors this behavior by transitioning at around 30 labeled points, matching $K = 10$ in performance after both become fully exploitative.

On the more complex EMNIST dataset, $K = 50$ outperforms $K = 10$ because more exploration is required. Our curvature-based method adapts to this complexity and ultimately outperforms both schedules after transitioning to exploitation. Interestingly, both our method and the $K = 50$ decay schedule set $\tau = 0$ at *the same point* - after 100 labels, on average. Despite this, our method achieves better downstream exploitative AL results, illustrating that the quality of exploration - not just its duration - matters. This suggests that a sharp transition from pure exploration to pure exploitation can be more effective than gradually decaying, as reflected in the marked jump in accuracy - consistently observed across all datasets - once τ is set to zero.

These results underscore the effectiveness of our curvature-based τ scheduler, which provides a lightweight and principled alternative to the K-based decay schedules that may not generalize well. Our method only switches to exploitation once sufficient coverage has been achieved, requires no hyperparameters, and adapts automatically to the graph's complexity, making it particularly suitable for real-world applications where class structure or label budgets are unknown in advance.



(a) MNIST (b) FashionMNIST (c) CIFAR-10

Figure 3.8: Comparison of sequential active learning performance with GBSSL variants on MNIST, FashionMNIST, and CIFAR-10. We start with 1 label per class and then run 100 AL iterations. We report mean (solid line) and standard deviation (shaded) over 100 trials. Hypergraph learning is a better classifier than Laplace learning, but manipulating the higher order weight matrix is exceedingly costly (see Tables 3.2 and 3.3). Our proposed AL Local Rewiring method strikes the best balance of accuracy and efficiency by only computing and storing the higher order terms at active learning acquisitions. [HMW25, Figure 8]

3.4.5 Localized Graph Rewiring

In this section, we evaluate the performance of our method outlined in Algorithm 5. Our method, denoted AL Local Rewiring, is compared against three baselines:

- Hypergraph: the standard Hypergraph approach (3.3.3), using full powers of the Laplacian on the entire graph,
- Laplace: using only the standard graph Laplacian L_{k_1},
- Random Local Rewiring: a control strategy in which additional regularization is applied around randomly selected nodes, rather than the nodes acquired by the active learning policy.

We set $q = 2$, $k_1 = 50$, $k_2 = 30$, with powers $p_1 = 1$, $p_2 = 2$, and quadratic weights $\lambda_1 = 1$, $\lambda_2 = 4$, following the multiscale design of [WBT25a]. Figure 3.8 visualizes the

Method	MNIST		F-MNIST		CIFAR-10	
	Acc. (STD)	Time (s)	Acc. (STD)	Time (s)	Acc. (STD)	Time (s)
AL-LR	69.84 (11.29)	640	47.57 (11.75)	731	58.76 (11.11)	378
HG	90.40 (7.70)	9441	61.74 (6.40)	10720	66.96 (11.09)	4623
R-LR	55.63 (12.69)	627	40.96 (11.00)	753	53.49 (12.21)	416
Laplace	59.38 (12.49)	546	39.24 (10.40)	656	53.14 (10.99)	349

Table 3.2: Accuracy and Time for 50 AL Iterations (60 total labeled points). We report means and standard deviations over 100 trials. The methods are AL Local Rewiring (AL-LR), Hypergraph (HG), Random Local Rewiring (R-LW), Laplace. [HMW25, Table 2]

performance of each method throughout the AL process over 100 trials, and Tables 3.2 and 3.3 summarizes results and provides efficiency details.

The standard hypergraph approach, which applies full Laplacian powers over the entire graph, achieves the highest overall accuracy. This confirms the theoretical benefits of rich multiscale regularization when computational cost is not a limiting factor.

Our proposed method, AL Local Rewiring, consistently ranks second and yields substantial improvements - typically 5–10% over classical Laplace learning. These gains are consistent across all datasets and especially pronounced in the early rounds of active learning, where label scarcity hampers traditional methods. This demonstrates the effectiveness of localizing higher-order regularization around informative nodes. In contrast, the Random Local Rewiring control performs nearly identically to Laplace learning. This suggests that simply adding higher-order smoothing is not sufficient - targeted application around informative nodes is crucial.

In terms of efficiency, AL Local Rewiring matches the runtime of Laplace learning while avoiding the substantial overhead of the "full" hypergraph method, which is approximately ten times slower in our experiments. AL Local Rewiring therefore captures most of the

Method	MNIST		F-MNIST		CIFAR-10	
	Acc. (STD)	Time (s)	Acc. (STD)	Time (s)	Acc. (STD)	Time (s)
AL-LR	95.37 (2.83)	1273	65.65 (9.51)	1439	75.96 (7.28)	744
HG	96.47 (1.59)	18376	71.77 (4.58)	20340	77.18 (7.03)	8908
R-LR	93.33 (4.86)	1246	63.09 (10.55)	1449	72.45 (10.48)	821
Laplace	94.79 (3.34)	1072	60.44 (10.23)	1262	73.28 (8.65)	683

Table 3.3: Accuracy and Time for 100 AL Iterations (100 trials). The methods are AL Local Rewiring (AL-LR), Hypergraph (HG), Random Local Rewiring (R-LW), Laplace. [HMW25, Table 3]

performance benefits of full multiscale regularization at a fraction of the computational cost. Overall, our proposed strategy strikes the best balance of accuracy and efficiency by only computing and storing the higher order terms at active learning acquisitions. These results support the use of AL Local Rewiring as a scalable alternative to full multiscale regularization, particularly in real-world settings where efficiency is critical.

3.5 Conclusion and Future Work

This chapter introduces methodological and empirical advances that leverage graph-topological tools to improve both accuracy and efficiency in the low-label regime. First, we propose a novel coreset selection algorithm, Curvature Coreset (CC), which uses Balanced Forman Curvature to select representative labeled nodes that capture the graph's underlying cluster structure. We further demonstrate that BFC serves as an effective signal for determining when exploration is complete, providing a principled stopping criterion and enhancing existing AL routines such as PWLL-τ. Empirically, this leads to improved classifier initialization and consistently outperforms baseline strategies across multiple benchmark datasets in both sequential and batch AL settings. Together, these results establish curvature as a powerful

mechanism for addressing the central *exploration–exploitation trade-off* in graph-based AL.

Complementing this, a second focus of our work shows that *modifying* the graph topology yields substantial performance gains. In particular, we introduce a localized multiscale regularization technique that selectively enhances structure around labeled nodes. This approach improves over standard Laplace learning in accuracy, while achieving speedups of over an order of magnitude compared to full hypergraph-based multiscale methods.

There are several promising directions for future work. First, a more principled understanding of how graph properties—such as homophily, degree distribution, or curvature—influence the exploration–exploitation trade-off could enable adaptive acquisition strategies tailored to specific graph regimes. For instance, we hypothesize that highly homophilous graphs may require fewer exploratory labels, while heterophilous graphs may benefit from broader initial coverage. "Heterophily-aware" models have already found success in graph attention networks [WGY24]. Second, curvature-based techniques open the door to new directions in large-scale and practical active learning settings. For batch active learning, curvature-driven alternatives to LocalMax [CCT23] - a method that extends the acquisition process to batches by relying solely on neighbor information - could better reflect the graph topology, enhancing data efficiency and ensuring balanced exploration across diverse regions of the graph. For graph reduction, aimed at enabling efficient learning on very large datasets, curvature-informed grouping criteria could yield geometrically consistent reductions while preserving key structural features. Finally, our use of BFC highlights the untapped potential of integrating tools from the GNN literature into GBSSL. While GNNs excel in flexibility, scalability, and empirical performance on large datasets, GBSSL offers a mathematically tractable framework that can succeed with minimal labeled data. Techniques such as curvature-based metrics, rewiring strategies, and spectral objectives - typically used to improve message passing in GNNs - may have powerful analogues in GBSSL, offering new opportunities for both algorithm design and theoretical insight. Exploring these connections could lead to unified frameworks that bridge geometric, spectral, and information-theoretic

r	Time (s)
1	1908
10	195
50	44
100	25
200	15

Table 3.4: The reduction parameter's effect on time to acquire 100 points on CIFAR-10. [HMW25, Table 4]

perspectives on label-efficient learning.

3.6 Data Availability

The datasets used in this chapter are either freely available online and cited in the Chapter, or available on our GitHub repository `https://github.com/hardiman-mostow/TopologyActiveLearning`.

3.7 Appendix

3.7.1 Reduction Parameter

In order to speed up CC, we proposed a "reduction parameter" r, which reduces the search space by a factor of r by only considering the top $1/r$ nodes by degree (see Remark 3.3.3). This aligns with the definition of BFC (Definition 3.2.2) which is already biased toward high degree nodes. Figure 3.9 illustrates (on CIFAR-10) that this reduction parameter has no effect on the accuracy of CC, so long as r is not too large. For example, $r = 50$ provides almost identical accuracy, while being about 50 times faster. Increasing r beyond that appears to reduce the search space too much, causing worse performance. We also give

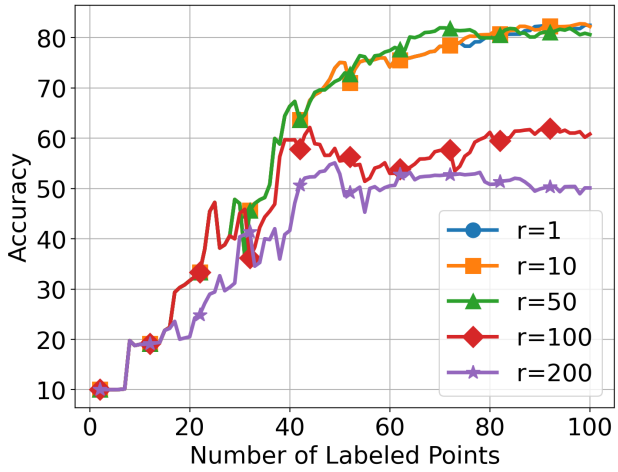


Figure 3.9: Comparison of reduction parameter r values for CC on CIFAR-10. r reduces the search space and helps speed up CC without sacrificing accuracy, up to a certain threshold where the search space becomes too small. [HMW25, Figure 9]

timing details in Table 3.4.

3.7.2 DAC Coreset Sizes

One disadvantage of DAC is the high variability of coreset sizes. Due to the stochasticity of the algorithm, coreset sizes differ between trials even with a fixed radius R. This means that DAC may produce drastically different results, even for the same data and the same radius. This also could pose problems in real-world settings where labeling budgets may be fixed. To illustrate this, and to provide transparency in our experiments, Table 3.5 reports DAC coreset statistics on the image datasets. Even for a fixed radius, DAC produces drastically different coreset sizes. For example, on FashionMNIST, a radius of 0.4 results in coreset sizes ranging from 89 to 159.

Dataset	**Target Coreset Size**	**Radius**	**Min**	**Mean**	**Max**	**Std**
MNIST	50	0.25	47	49.70	56	2.53
MNIST	100	0.125	96	104.80	140	12.13
FashionMNIST	50	0.4	47	53.60	76	7.88
FashionMNIST	100	0.2	89	102.80	159	21.26
CIFAR-10	50	0.35	50	55.80	61	3.22
CIFAR-10	100	0.175	92	100.50	104	3.44

Table 3.5: DAC Coreset Sizes Across 10 Trials. [HMW25, Table 5].

CHAPTER 4

Deep Self-Supervised Disturbance Mapping with SAR Data

This Chapter is a version of [HMH25], published in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS) and adapted here under a CC BY 4.0 license. This is joint work with Charlie Marshak and Al Handwerger, who were my mentors during an internship at NASA Jet Propulsion Lab (JPL). Charlie and Al proposed the internship project and provided the data. I led the ideation, developed and implemented the model, ran the experiments, and wrote the paper as first author under their supervision.

Real-time disturbance maps are a critical tool for first responders and civilians during and after natural disasters. Satellite remote sensing provides an abundant data source for identifying disturbances; however, hand-labeling datasets is impractical given the constant stream of new imagery. In this Chapter, we design and train a self-supervised vision transformer to map land disturbances utilizing a newly available, near-global, analysis-ready synthetic aperture radar (SAR) dataset. The model is more operationally practical than prior work, requiring no labels to train and generalizing effectively without any retraining or finetuning. When evaluated on several real-world natural disaster events of varying terrain and scale, our model outperforms existing SAR-based methods across metrics, advancing deep learning-based SAR disturbance mapping toward real-world deployment.

4.1 Introduction

The Earth's surface is continually changing due to a combination of natural [MBG12, LM12, Pel04] and human-induced factors [CSH18, HPM13]. While some changes occur gradually over time (e.g., mountain formation, glacial isostatic adjustment), others are abrupt and can have immediate impacts on the environment (e.g., landslides, wildfires, flooding, deforestation, mining). Satellite remote sensing can detect these surface changes and disturbances over large areas and on various time scales to address critical environmental and societal challenges [RBE15, KH13]. Through disturbance delineation, we enhance our capability to monitor vulnerable or protected habitats [RBE15], map the impacts of human urbanization and development [AKK20], track population migration patterns [MT92], identify tropical forest loss [HPM13, RMS21], and respond to natural hazards such as earthquakes [SKZ21], floods [PHV21, BTA20], landslides [HHJ22, AKS21], and fires [BZN20, RV93].

Among remote sensing techniques, synthetic aperture radar (SAR) is an invaluable modality for disturbance mapping, with the capability to rapidly image the ground regardless of cloud cover or illumination conditions. While optical imagery provides information about *spectral* changes, SAR provides information about *structural* changes that are captured via backscatter and phase. Applications include centimeter-scale motion of the Earth's surface [RHJ00, BW20], crop mapping [HZG23], structural damages after earthquakes [SKZ21], and landslide [JY20, JYK17], and wildfire delineation [RV93, BZN20].

Despite the proven capability of SAR for disturbance mapping, analysis-ready SAR data has - until recently - not been immediately available, requiring expert processing to produce such datasets over large areas and timescales. However, the release of the near-global Radiometric Terrain Corrected SAR backscatter from the C-band Sentinel-1 (RTC-S1) dataset by NASA's Observational Products for End-Users from Remote Sensing Analysis (OPERA) project in October 2023 has effectively removed this barrier for Sentinel-1. OPERA RTC-S1

provides publicly available dual-polarization[1] SAR backscatter imagery in an analysis-ready format [BBC23]. Moreover, RTC-S1 mitigates the effects of terrain and SAR acquisition geometry, such as layover and shadow, that can obfuscate surface changes observed by SAR [Sma11, SFM23]. These three qualities - analysis-ready, near-global, and terrain correction - makes OPERA RTC-S1 an ideal dataset for systematic and thematic disturbance delineation.

The aim of this chapter is to train a *self-supervised*, transformer-based model capable of probabilistically delineating *generic* disturbances in OPERA RTC-S1 time-series. Here, *generic* disturbances refers to any changes that deviate from the set of baseline imagery used to establish a range of nominal observations of the land surface.

Supervised machine learning techniques rely on the existence of a large, labeled training set. However, labeling such datasets for disturbance mapping - particularly global-scale SAR time series spanning months or years - is impractical: it requires expertise of SAR physical mechanisms [Woo17, OQ04] as well as knowledge of the land cover being monitored to determine what signals are phenological and which are true disturbances. Despite this, many previous works on SAR disturbance mapping are supervised [BZN20, VMB23, NMC21, NMB24], requiring carefully curated, labeled training sets. Not only is this difficult to scale to larger datasets, but the models trained on these specialized datasets lack the generality to be a broadly applicable, generic disturbance model. While there are numerous optical disturbance datasets [CS20b, JWL19, IBM24, RAK20b] and corresponding supervised models (see e.g. [CSH24]), training a SAR model on optically-derived labels presents significant challenges as some disturbances visible to an optical sensor will be invisible to SAR and vice versa. Additionally, with forthcoming NASA ISRO SAR (NISAR) data, where the creation of a large labeled corpus from the new L-band imagery is not possible, the demonstration of a label-free approach for SAR-based applications is essential.

[1]The two polarizations are co-polarized and cross-polarized, also known as vertical transmit and vertical receive (VV) and vertical transmit and horizontal receive (VH).

This motivates *self-supervised* learning for SAR-based disturbance mapping. Self-supervision remedies the need for labels by generating a supervised signal from the data itself [HCX22, MLN21, CKN20]. One previous performant self-supervised work on SAR damage mapping was by Stephenson et al. [SKZ21], which utilized a recurrent neural network (RNN) [RHW86] on interferometric SAR coherence to map earthquake damage extents (conversely, this chapter considers SAR backscatter). RNNs are neural networks designed to handle temporal data, such as language processing tasks [Sut14]. However, more recently, RNNs have largely been supplanted by transformers [Vas17], which are now state-of-the-art in a variety of tasks, including language [BMR20] and vision [DBK20], as well as remote sensing tasks [CKM22, TCZ23]. Moreover, transformers avoid exploding and vanishing gradients, which tend to plague RNNs. Hence, this chapter will train a transformer in a self-supervised fashion. We also improve on [SKZ21] by training and deploying a single model on multiple types of damage events in three different parts of the world, whereas [SKZ21] trained separate models for each location, and only focused on earthquake damage extents.

4.1.1 Our Contributions

in this chapter, we train a self-supervised vision transformer [DBK20] on near-global OPERA RTC-S1 imagery and demonstrate superior damage mapping performance across environments and disaster events compared to previous works. Our transformer learns to estimate the per-pixel distribution from baseline of OPERA RTC-S1 imagery. This estimated distribution can be compared to a new acquisition by computing the Mahalanobis distance, yielding a disturbance metric that can be used to delineate disturbance. The metric has a statistical interpretation as the number of standard deviations (SD) from the modeled per-pixel mean. More simply put, the metric quantifies how likely a pixel is disturbed, with larger values indicating greater likelihood of disturbance. The vision transformer is trained in a self-supervised manner, meaning no labels or human annotation are required; the only input is the OPERA RTC-S1 imagery. We evaluate our disturbance metric on external validation

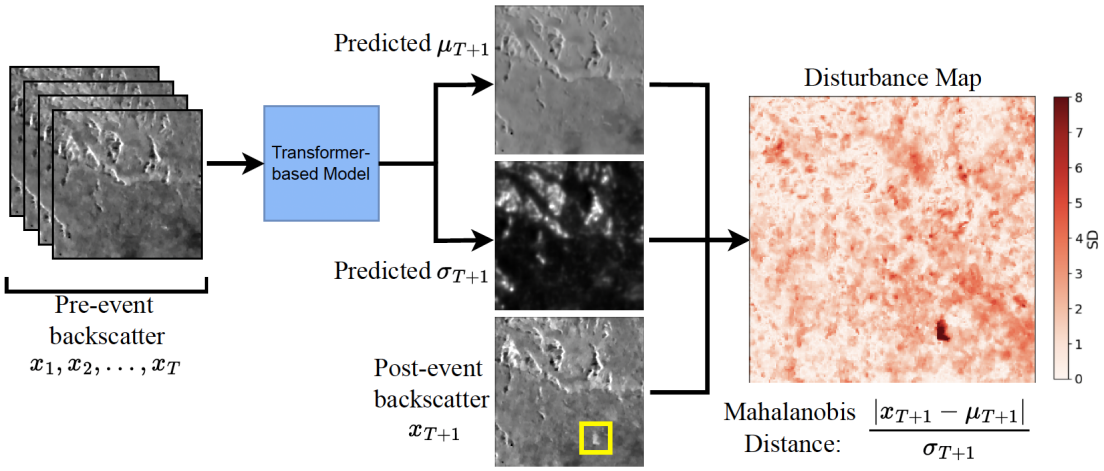


Figure 4.1: Visualization of the disturbance metric and how it is computed for single channel (VV) time-series. This example shows data from a landslide that occurred in Enga Province, Papua New Guinea on May 24, 2024. We use a sequence of baseline imagery $x_1, x_2, ...x_T$ (left) from before the event to predict a distribution for each pixel in the image with mean μ_{T+1} and standard deviation σ_{T+1} (center right). These parameters are used to quantify disturbance in the observed image x_{T+1} (center right, bottom) to compute one-dimensional transformer metric via the Mahalanobis distance (right), where higher values (measured in standard deviations or SD) indicate a higher likelihood of disturbance. Our methodology is able to precisely delineate the landslide disturbance extents (bottom right corner of the Disturbance Map). [HMH25, Figure 1]

data from three recent natural disasters: 1) a landslide in Papua New Guinea, 2) a series of fires in Chile, and 3) flooding in Bangladesh. We use the same model and metric for every event. The validation data is derived from publicly-available damage maps [Hum24, Cop24]; much of which is made from Very High Resolution (VHR) optically-derived inputs. These natural disasters represent high-intensity disturbances where we expect significant alignment between optical- and SAR-based disturbance delineations. The validation data used for this chapter, including the disturbance delineations and data provenance, are found in the GitHub repository [OPE24a]. We demonstrate that the proposed vision transformer-

based disturbance model has systematically better performance compared to an RNN-based model [SKZ21] and the classical log ratio method for SAR change mapping [RV93]. Code for our data[2] and experiments[3] are available via GitHub.

4.1.2 Organization of Chapter

The remainder of this Chapter is organized as follows. In Section 4.2, we review previous works on disturbance mapping with SAR, transformers, and self-supervision. In Section 4.3, we detail the mathematical formulation of our model, our disturbance metric for computing delineations, and the model architectures and training. Sections 4.4 and 4.5 describe the OPERA RTC-S1 data (and external validation data) we use to train and evaluate our model, respectively. Section 4.6 presents our disturbance mapping results on three natural disasters, where our transformer-based model consistently outperforms previous works (RNN and Log Ratio). We also present ablation experiments that justify our model choices. Although our evaluation is limited to three disasters, the ability to accurately delineate different categories of disturbances in varied environments using a single vision transformer provides some encouraging evidence that such an approach may be applicable to monitor disturbance at larger scales. To this end, we will discuss future work to further evaluate this model in Section 4.7 and 4.8.

4.2 Background

4.2.1 Disturbance Mapping

Disturbance mapping is the process of identifying pixels in space and time that undergo changes (i.e. disturbances) within a coregistered time-series of remotely-sensed imagery.

[2] `https://github.com/OPERA-Cal-Val/dist-s1-events`

[3] `https://github.com/opera-adt/distmetrics`

Disturbance mapping is sometimes referred to as "change detection" in the remote-sensing literature [Can19, ZTM17]. While disturbance mapping encompasses many types of changes, our work will focus specifically on abrupt, sudden disturbances (i.e., damage) caused by natural disasters. Natural disasters are a key application area for disturbance models and simplify validation by providing a clearly defined point in a time series to map disturbances. Hence, we will use "change", "damage", and "disturbance" interchangeably. in this chapter, we are focused on *generic* Sentinel-1 disturbance delineation - that is, modeled statistical deviation of the current acquisition from a set of baseline images. Generally, what disturbances can be mapped depends on the temporal sampling of inputs, the spatial resolution, the type of change being monitored, the modality of the sensor, and the data used to construct the baseline. Sentinel-1 data, and hence OPERA RTC-S1 data, currently has 12-day sampling frequency[4], though not all Sentinel-1 passes result in new data [Ala24]. While this means some disturbances may not be resolved by Sentinel-1 (a flash flood can occur and resolve between samples), our approach leverages the available data's strengths and is applicable to *any* time series generated by Sentinel-1.

A similar view of generic disturbance is found in OPERA's near-global *optical* Land Surface Disturbance from Harmonized Landsat Sentinel-2 (DIST-HLS) generic disturbance algorithm [BBC23, HPS24]. The DIST-HLS algorithm estimates the per-pixel mean and standard deviation using sample statistics from multiple monthly composites and measures deviation from the baseline via the Mahalanobis distance [HPS24]. in this chapter, we develop an algorithm that provides a complementary measure of generic disturbances using SAR, with the future goal of deploying it at the near-global scale.

Currently, there are two operational[5] SAR disturbance products, which track tropical

[4]Sentinel-1 A/B constellation had a 6 day repeat pass frequency. Since Sentinel-1 B's decommission in December 2021, Sentinel-1 A is the only satellite collecting data and has 12 day repeat pass frequency. When Sentinel-1 C is launched in December 2024, the constellation will again have 6 day temporal sampling.

[5]Here, operational here means that each new SAR acquisition triggers the generation of a new disturbance

forest loss: Advanced Land Observing Satellite (ALOS) forest alerts [KWH19, RMS21] and RAdar for Detecting Deforestation (RADD) [RMS21, RVV18]. Both rely on estimating backscatter conditioned on "forest" and "non-forest" distributions and delineating disturbance when there is evidence forest has been removed based on these modeled distributions. Our approach aims to delineate a broader class of disturbances, across terrains and disturbance events. To this end, we utilize a vision transformer trained on near-global RTC-S1 data to determine the range of nominal backscatter values in varied environments and land cover types, not just dense tropical forest.

In order to contextualize our approach, it is helpful to define the notion of a "disturbance metric." A disturbance metric compares a baseline set of images to a recent image acquisition to determine the per-pixel likelihood of disturbance [DDT12]. Larger values of the metric typically indicate greater likelihood of disturbance. Using a disturbance metric, delineation of disturbances (that is, classifying each pixel as "disturbed" or "undisturbed") is simplified to thresholding the metric. For example, the classical log-ratio method [RV93] defines a disturbance metric that identifies changes based on the decibel (dB) deviation from the baseline image set:

$$\ell := \left| \log_{10} \left(\frac{I_1}{I_0} \right) \right| = \left| \log_{10}(I_1) - \log_{10}(I_0) \right| .$$

Above, I_1 is the post-event cross- or co-polarization image and I_0 is a pre-event cross- or co-polarization image. I_0 is also sometimes an aggregated reference image; in our experiments, we will set I_0 to be the median of the available pre-event imagery. Although it was developed over 30 years ago, log ratio is still widely used in modern SAR damage mapping research [ASK22, HHJ22, JY20, LHW22, MSR19, MGC21, PLK24]. It is well known that large negative values in $\log(I_1) - \log(I_0)$ indicate vegetation loss when we consider cross-polarization data [RV93, Woo17]. Above, we consider the absolute value because we aim to delineate generic disturbances from a baseline set of imagery, which could be indicated by

map.

anomalously low or high measurements in either polarization. In Section 4.3.2, we will define a disturbance metric for our transformer-based approach, which compares favorably to ℓ in our experiments (Section 4.6).

Similarly, one can fashion likelihood tests to generate disturbance metrics [DDT12, ZDD19]. Although a disturbance metric is a per-pixel measure, it is often helpful to incorporate nearby pixel information to better contextualize disturbance e.g. through superpixels [MSD19], convolutional layers of a neural network [BSB19], or Markov random fields [BP00]. in this chapter, our vision transformer is designed to estimate the per-pixel distribution from the baseline set using rich spatio-temporal information extracted from the data via the neural network architecture. We utilize the learned distributional parameters to construct a disturbance metric via the Mahalanobis distance between this estimated distribution and the most recent image. This metric has a convenient statistical interpretation as the number of (estimated) standard deviations away from the (estimated) mean at each pixel.

The goal of our disturbance algorithm is to develop a methodology that can identify disturbance with each new Sentinel-1 acquisition, utilizing the preceding acquisitions to establish a baseline [DHA20, RVV18, RMS21, HHJ22]. While this process is well-suited over large areas and multiple environments, there are also change detection methods focused on bi-temporal comparisons, i.e. using a single image pair, in which an abrupt disturbance and often damages from a natural disaster can be observed [CRO24, MBZ18, JWL19, CS20b, BBM05, JY20, YHO15]. Some algorithms, such as the proposed NISAR CuSum algorithm [NIS18, NIS22, SKL24], analyze a full year of imagery and then identify the changed areas in space and time from the available data. Our transformer-based approach can use anywhere from 2-10 acquisitions; all of our tests used at least 4 acquisitions (representing approximately a six week period, see 4.6.4).

Our work utilizes OPERA RTC-S1, a near-global dual-polarization backscatter product. Backscatter amplitude is one component of SAR remote sensing. The interferometric phase

and coherence are invaluable for damage assessment and disturbance [SKZ21, JKL16, JY20]. While incorporating such data into our models would provide richer structure, there is not a near-global interferometric phase product. The OPERA Coregistered Single Look Complex dataset (OPERA CSLC-S1), which contains phase information from Sentinel-1, is only available over North America. Even this higher-level product would require significant preprocessing to incorporate into our model and such processing is beyond the current scope of this chapter.

4.2.2 Transformers and Self-Supervised Training

Transformers [Vas17] are becoming the de-facto architecture across numerous machine tasks, producing state-of-the-art results in language modeling [OA24, Dev18], image classification [DBK20,HCX22], and semantic segmentation in imagery and video [HCX22,KMR23,FFL22], among other areas. Empirically, transformers are highly generalizable and can be fine-tuned on downstream tasks with relatively small labeled datasets [HCX22, KMR23, BMR20]. Transformer's ability to learn new tasks quickly has given rise to "foundation models" [YCC21, LLR24]. These models are becoming standard in geospatial data analysis [CKM22, TCZ23, RGL23, JRP23, Cla24, SLC24]. These geospatial transformers too can be fine-tuned across many standard remote sensing analyses including delineation of flood extents, wildfire perimeters, and landslides [JRP23], estimation of biomass, agricultural yield, and tree height [Cla24, TCZ23, CCG24], and generation of land-use maps at the continental scale [ADG23]. Typically, these models are trained in a self-supervised fashion, meaning that no labeled data (or very limited labeled data for fine-tuning) is required to produce results.

Self-supervised learning is a training methodology in which a supervised learning signal is created by the input data itself. There are numerous approaches to self-supervised training including autoencoding [KW13], residual learning/backward diffusion [ZZC16, HJA20], masked autoencoding [HCX22, TSW22, FFL22, BMR20] and contrastive learning [CKN20]. In remote sensing, self-supervised training is invaluable as it removes the requirement to

create enough labels to train deep neural network models. Moreover, it allows these deep models to be quickly adapted to various sensors, each of which may have unique sensitivities, varying acquisition patterns, and often large unlabeled data archives. Indeed, self-supervised learning is becoming the standard way to train these transformer-based models for geospatial analysis [RGL23, TCZ23, MLN21, KAM24, RGL23, ADG23, Cla24, JRP23, LLR24]. Our self-supervised approach will follow the method proposed in [SKZ21], where our models will predict a pixel-based distribution for an image and use a statistical measure (z-score) to map disturbances. The model used in [SKZ21] is an RNN, focuses only on disturbances caused by earthquakes, and trains a new model for each earthquake-affected area studied. We will show that the *transformer*-based architecture improves disturbance delineation over an RNN for the OPERA RTC-S1 data, and that a *single* transformer-based architecture can be used across multiple environments and across multiple categories of disasters.

We will briefly review some key components of transformers. At the heart of transformers is the attention mechanism, which computes an attention "score" based on three learned matrices: the query Q, the key K, and the value V.

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V, \tag{4.2.1}$$

where d_k is the dimension of Q and V. Attention allows the model to learn how different elements of a sequence "attend" to one another, learning complex temporal relationships without the exploding or vanishing gradient issue of RNNs. Another novelty of transformers is their positional encoding (these can be either learned or fixed - we will use learned), which is added to the input embeddings to indicate position to the model. Combined with residual connections [HZR16a] and layer normalization [LKH16], transformers have been shown to be a powerful tool in machine learning.

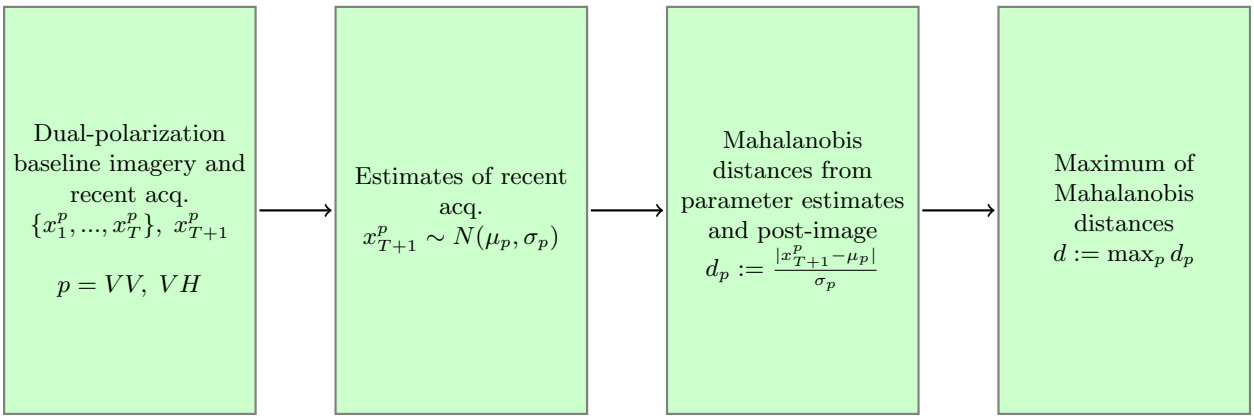


Figure 4.2: A flow chart illustrating the transformer metric proposed in this chapter. The metric is computed for dual polarization images (see Fig. 4.1 for 1-dimensional visualization). The first arrow is where the deep-learning model is utilized to estimate the per-pixel distribution of the x_{T+1}. [HMH25, Figure 2]

4.3 Methodology

4.3.1 Mathematical Formulation of the Model

Let $X := \{x_1, x_2, ..., x_T, x_{T+1}\}$ be a sequence of coregistered SAR images, with each $x_i \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels and H and W are the height and width in pixels, respectively. We call the images $x_1, ..., x_T$ the set of baseline imagery, or pre-images. The image x_{T+1} represents the latest acquisition image, or post-image. The input to the model is the set of baseline images, and the output is an estimate of the per-pixel distribution of x_{T+1}. For our work, the channel dimension C will be 2 as our input imagery is dual-polarization, i.e. VV and VH (co-polarized data VV is vertical transmit and vertical receive and cross-polarized data VH is vertical transmit and horizontal receive). The spatial dimensions are $H = W = 16$.

Let f be the model parameterized by weights θ. Our model seeks to learn the parameters of a normal distribution at each pixel at timestep $T + 1$ from the set of baseline imagery. More precisely:

$$\mu_{T+1}, \sigma_{T+1} = f_\theta(x_1, x_2, ..., x_T), \tag{4.3.1}$$

where $\mu_{T+1}, \sigma_{T+1} \in \mathbb{R}^{C \times H \times W}$. Since our forecast is probabilistic in nature, we minimize the negative log-likelihood loss to find the optimal parameters θ^* on samples from our training set D, consisting of sequences of variable-length baseline imagery:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{X \in D} \Big(\frac{1}{2} \log(2\pi\sigma_{T+1}^2) + \frac{(x_{T+1} - \mu_{T+1})^2}{2\sigma_{T+1}^2} \Big). \tag{4.3.2}$$

This loss assumes (a) that each polarization is independent and (b) that each pixel in the image x_{T+1} is independent. Assumption (a) is made because the anti-diagonal components of the covariance matrix of the dual-polarization data typically are small and thus numerically unstable, in part because the two polarizations are correlated over some land cover types and anti-correlated in others. Assumption (b) is made so we can write down the negative log-likelihood for each training sequence coming from D at once [SKZ21]. We note that, while we assume each pixel in $T + 1$ is independent, all the pixels in the set of baseline images $x_1, x_2, \ldots, x_T$ are used in any given estimate of a pixel's distribution in x_{T+1}. Equation 4.3.2 can be optimized using any standard optimization routine; we use the Adam optimizer [KB14]. Note that the loss is computed with respect to the $T + 1$ image, and requires no label information; the training process is entirely self-supervised.

4.3.2 The Disturbance Metric and Disturbance Mapping

In the previous subsection, we detailed how to take a sequence of baseline images $x_1, ..., x_T$ to estimate the per-pixel distribution. We can now define a disturbance metric, d_p, that provides a per-pixel measurement of disturbance by comparing the estimated distribution to the most recent acquisition x_{T+1}. Consider a single polarization p, p = VV or VH. We

define our disturbance metric as the 1-dimensional Mahalanobis distance:

$$d_p = \frac{|x^p_{T+1} - \mu_p|}{\sigma_p}.$$

This metric can be interpreted as the number of standard deviations from the estimated mean x_T. The larger d_p is, the more likely disturbance has occurred, where disturbance is with respect to the baseline imagery. A similar metric was used in [SKZ21], where the absolute value is omitted and referred to as the z-score; however, that work studied interferometric coherence, where a decrease is most important for damage assessment, and so sign must be preserved. In our case, we consider absolute deviation, since increases or decreases in backscatter could indicate disturbance.

We then combine each polarizations taking the maximum of d_p, i.e. $d = \max_p d_p$. Taking the maximum over d_p effectively means that our delineations consider disturbances in *either* polarization, VV or VH. One could also consider combining the metrics via a minimum (analogous to a logical "and" over the channels) or via addition (a 2-dimensional Mahalanobis distance assuming channel independence). We elected the maximum (analogous to a logical "or") as disturbances are often more detectable in one channel than the other. Indeed, many disturbance related works focus on a single polarization [NBM22, RV93].

A summary of our approach is shown in Fig. 4.2. To map disturbance, we threshold $d > \tau$ for some positive real number τ. For the rest of this chapter, we call d the *transformer metric*. Our metric has a clear probabilistic interpretation that a metric exceeding some τ has an associated probability. For example, if we assume normality of a pixel, we know that $d_p > 3$ occurs less than 1% of the time. In our disaster data, we found thresholds τ in the range of 3-5 visually suitable for disturbances delineations. We will systematically analyze thresholds τ using optically-derived validation data in Section 4.6.

Throughout our experiments, we will compare to an RNN-based model with the same disturbance metric, and a log ratio-based metric using ℓ in Section 4.2.1, which takes the absolute deviation in dB with respect to a particular polarization. We again combine polar-

izations via the maximum and set a baseline I_0 by taking a per-pixel median of the pre-event imagery.

4.3.3 Model Architecture, Hyperparameters, and Training

We use an encoder-based vision transformer architecture. We take patches of the input images and add learned spatial and temporal positional embeddings, as done previously for spatiotemporal data in [TSW22, FFL22, CKM22]. We elected not to use more sophisticated spatial-temporal encodings (see e.g. [TCZ23, RGL23, Cla24, JRP23]) as we wanted our model to be simple to illustrate the efficacy of this type of approach to disturbance mapping. We will explore more sophisticated spatiotemporal positional encoding in the future.

Our transformer encoder uses a model dimension of 256, 4 attention heads, 4 layers, and a feedforward dimension of 768. The input to the model is a sequence of T SAR images, each with size $2 \times 16 \times 16$ (channels, heigh, and width, respectively, for a total input size of $T \times 2 \times 16 \times 16$. The channel dimension 2 is due to the dual-polarization SAR imagery (VV/VH). In each batch, T is chosen randomly to be between 2 and 10 so the model can learn to forecast the distributions with varying amounts of pre-event information. Operationally, this is important as data may often be missing or an area of interest lacks a long acquisition history. For example, Sentinel-1 data does not consistently image in 12-day increments, so in some cases many months can go by before additional data is collected. Our validation datasets discussed in Section 4.6 have 10, 7, and 4 pre-event images available, underscoring the need for a temporally flexible model. Future work will focus on scaling up the model to allow for larger values of T.

Each $2 \times 16 \times 16$ SAR image is "patchified" into $2 \times 8 \times 8$ patches with a learned spatial and temporal embedding. For example, for a sequence of ten 16×16 SAR images, the input sequence to the transformer will have 40 (4 patches $\times$ 10 temporal steps) elements.

We use a dropout of 0.2 and ReLU activations. After the transformer learns representa-

tions of the input sequence, we use a 2 layer, fully connected (with hidden dimension 768 and a ReLU activation) projection head to output the predicted mean and standard deviation of each pixel in each channel in the subsequent image (both the mean and the standard deviation predictions have their own projection head). This results in a model with approximately 3.3 million parameters.

Previous work on deep learning-based disturbance mapping on SAR used an RNN Gated Recurrent Unit (GRU) [Cho14, SKZ21]. For a fair comparison, we construct a GRU with a nearly identical parameter count as the transformer (we note that the original work [SKZ21] applied a GRU to coherence values from SAR, whereas our work uses backscatter). To construct the GRU architecture with 3.3 million parameters, we use a model dimension of 326, 4 layers, a hidden dimension of 978 for the projection head, and the same dropout value, 0.2. This model also uses projection heads for the mean and standard deviation predictions. Since transformers utilize patches with positional embeddings, it is not obvious whether a fair comparison to an RNN should have an input size of 8×8 or 16×16. We found that the input size of 8×8 performed better (although still surpassed by the transformer), which is the model we report results for throughout Section 4.6. For completeness, we include results for the 16×16 input size in Section 4.6.6.

The training of both models use a batch size of 256, and train for 50 epochs using Adam [KB14] with an initial learning rate of 0.0001, which then decays to 0.00001 after 25 epochs. All experiments were run on a single NVIDIA A100 GPU with an 80GB memory.

4.3.4 Model Inference

For model inference, we often have much larger images than 16×16; a single burst RTC-S1 product is approximately 1000×3000 pixels. To apply our model, we sweep it across the image (for computational reasons, we use a stride of 4 for the fire and flood delineations, see Section 4.6) and average the estimates generated at each pixel. Without this sweeping approach, the predictions leave "edge artifacts" because the borders of each 16×16 window

have less neighboring pixels to inform the estimates. Indeed, inference of our small windows exhibit the so-called receptive field phenomena [LLU16].

4.4 RTC-S1 Data

The OPERA RTC-S1 provides measurements of the radar backscatter, γ^0, corrected for radiometry and topography [Sma11,SFM23]. All RTC-S1 products can be accessed through the Alaska Satellite Facility Distributed Active Archive Center (ASF DAAC) [OPE23]. These corrections result in an analysis-ready product that primarily captures signals related to the physical characteristics of ground-scattering surfaces, such as surface roughness, soil moisture, and vegetation structure. The RTC-S1 is generated from Copernicus Sentinel-1 (S1) interferometric wide (IW) single-look complex (SLC) data, provided by the European Space Agency (ESA). Sentinel-1 constellation can currently acquire data at 12 day repeat pass temporal frequency, i.e. with respect to a fixed orbit track. The ground sampling when independent of orbit track is increased to between 1 and 6 days; this combines ascending (satellite flying north and looking east) and descending (satellite flying south and looking west) geometries. The imaging mode and temporal coverage of Sentinel-1 data varies with location on Earth [Ala24]. The RTC-S1 product has a near-global geographic scope that covers all landmasses, excluding Antarctica, and its temporal frequency aligns with the availability of Sentinel-1A SLC data. RTC-S1 currently only includes data from Sentinel-1A satellite because OPERA started forward mode production in October 2023 after the loss of Sentinel-1B. Each RTC-S1 product corresponds to a single Sentinel-1 burst and is mapped onto standardized Universal Transverse Mercator (UTM) or Polar Stereographic projection systems.

RTC-S1 uses the Copernicus global 30 m (GLO-30) Digital Elevation Model (DEM) as the reference for topographic correction and geocoding. The data are in GeoTIFF format, and have a spatial resolution of 30 meters. The RTC-S1 data also includes layover

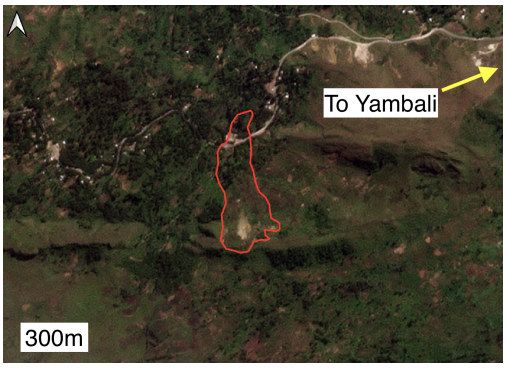
and shadow masks as well as bistatic and tropospheric corrections to improve data quality. Static layer files such as incidence angle, local incidence angle, masks, and number of looks are used. The OPERA RTC-S1 and static layers are generated using the InSAR Scientific Computing Environment version 3 (ISCE3) software developed at JPL/Caltech [RGA18]. Additionally, the RTC-S1 serves as the foundational dataset for other OPERA products including the Dynamic Surface Water eXtent from Sentinel-1 product (in production since Sep. 2024) and the future OPERA Disturbance from Sentinel-1 product (expected production in 2026). All of the OPERA products are designed to address the needs of U.S. Federal Agency stakeholders identified by the Satellite Needs Working Group.

4.4.1 Training Data

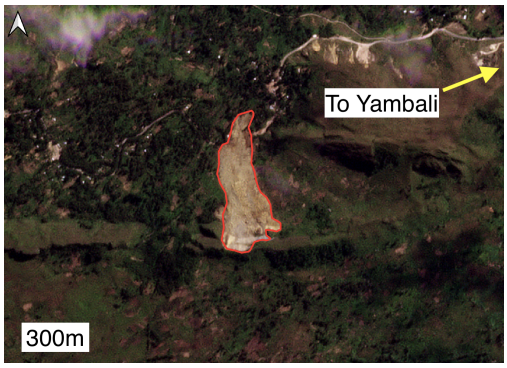
The training data is comprised of $2,511,348$ sequences of 11 SAR images from OPERA RTC-S1 (i.e., $T = 10$ is the largest possible value for T in training), which are each 2 channel (VV and VH) and 16×16 pixels. Future work will focus on increasing the size of the dataset and the temporal length of the sequences T. This training data is from OPERA RTC-S1 data that is currently not in the ASF archive, but rather stored in a public s3 bucket for the OPERA project validation [OPE24b]. The data is used to overlap the sites and validation period (October 2020 - October 2022) used by OPERA DIST-HLS validation [BBC23, PO24]. The data collection includes imagery from Sentinel-1B so some of the temporal sampling occurs with frequency of 6 days.

4.4.2 Data Preprocessing

Before using our model to estimate the distributional parameters, we perform basic SAR despeckling. We utilize homomorphic despeckling as described in [DDT17] using a total variational denoising model [ROF92]. Homomorphic despeckling is total variation denoising on dB-transformed backscatter imagery that, after despeckling, is inverted back into γ^0.



(a) May 15, 2024



(b) May 26, 2024

Figure 4.3: PlanetScope (optical) imagery [PBC] of before and after the landslide in Papua New Guinea. The red polygon shows the extent of the landslide (mapped manually by the authors). [HMH25, Figure 3]

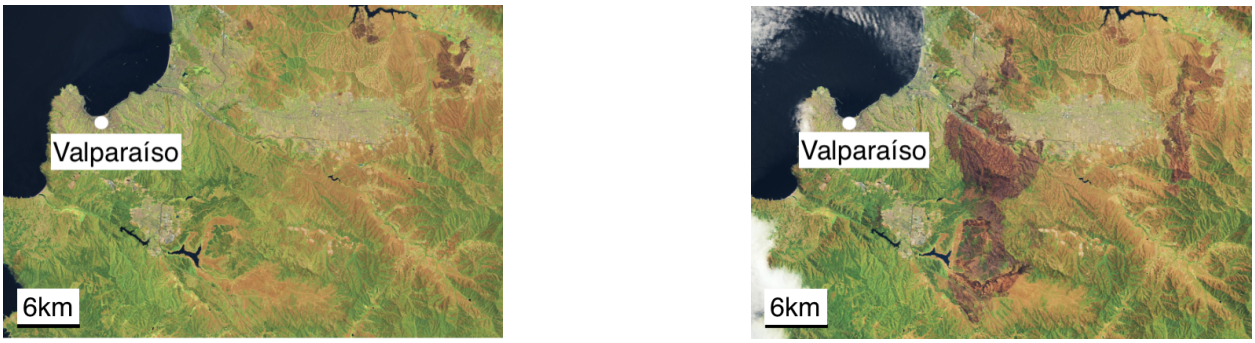
The probabilistic framework of our approach approximates the unknown pixel distribution using a Gaussian, which has domain $\mathbb{R}$. SAR γ^0 backscatter is within $[0, 1]$ so we follow the same pre-processing step as in [SKZ21], taking the logit transform of the backscatter data. The logit is a bijection of $[0, 1]$ to $\mathbb{R}$. This is done on the despeckled imagery.

$$\text{logit}(x) = \log\Big(\frac{x}{1-x}\Big) \tag{4.4.1}$$

For clarity, all figures in this chapter are presented in their original backscatter range $[0, 1]$. These two preprocessing steps are performed before application of the model described above.

4.5 Disaster Data

In this section we discuss the data used to evaluate our disturbance delineations. All the data was derived either manually by the authors (in the case of the Papua New Guinea Landslide) or by external disaster response projects such as UNOSAT or Copernicus Emergency Management Services (CEMS). We have curated these delineations and more events at this GitHub repository [OPE24a], along with provenance. The Bangladesh flood was derived



(a) December 20, 2023

(b) February 5, 2024

Figure 4.4: Landsat 8 and 9 false color imagery (bands 6, 5, 3) of before and after the fires in the Valparaíso region of Chile. The above imagery is taken from NASA Earth Observatory [NAS24]. [HMH25, Figure 4]

from Sentinel-1 data using an image pair by UNOSAT [Hum24]. The other disaster delineations (landslide and fire) are derived from VHR optical data (i.e. with spatial resolution $\leq$ 3m). Evaluating disturbance mapping with these ground truth datasets presents some limitations. First, we expect some inherent differences between disturbance maps derived from optical and SAR data. However, optical and SAR data tend to exhibit high agreement in these cases when dense vegetation is cleared by a fire or a landslide [Woo17]. Second, real disturbances unrelated to the natural disasters are not accounted for in this setting, but the abrupt nature of disasters means we expect this to have a negligible impact on our results. Overall, these natural disaster datasets serve as a high-quality ground truth for evaluating our methodology.

4.5.1 2024 Papua New Guinea Landslide

On May 24, 2024, a large catastrophic landslide occurred near Yambali, Enga Province, Papua New Guinea (see Fig. 4.3) [Pic24, ABC24]. News media reported that that the landslide buried numerous villages, with the death toll estimated to be between 650 and 2,000 people [Pic24]; the official death toll at time of writing is 670 [UN24]. In addition,

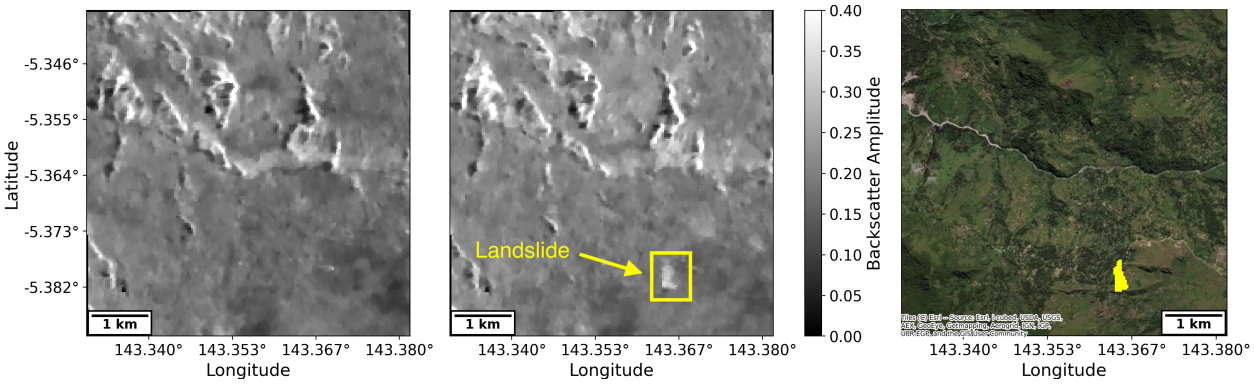


Figure 4.5: **Left**: Pre-landslide SAR imagery (VV), May 22 2024. **Middle**: Post-landslide SAR imagery (VV), June 3 2024. The landslide is seen in the bottom right corner. **Right**: Target (ground truth) damage map overlaid on ESRI Composite World Imagery [ESR24a]. The damaged area is shown in yellow. Note that the ESRI imagery is a composite and thus does not capture the recent landslide, and is used as a reference only. [HMH25, Figure 5]

approximately 7,000 people have been displaced by the event [Pic24].

Our input to the model is a sequence of ten pre-event OPERA RTC-S1 SAR images from 1/23/24 to 5/22/24 at 12 day intervals (except 5/10/24 when there was no data distributed despite a Sentinel-1 pass) and a post-event image from 6/3/2024. To delineate disturbances, we compute the Mahalanobis distance between the models' estimate distribution and this post-event image as described in Section 4.3.2. We analyze an area of approximately 6 km $\times$ 6 km. We manually mapped the landslide extent using PlanetScope optical imagery (3 m pixel) [PBC], see Fig. 4.5.

4.5.2 2024 Valparaíso, Chile Wildfire

In early February of 2024, a series of wildfires began in the Valparaíso region of Chile [Cas24]. Exacerbated by historic drought in the region and the 2023 El Niño weather event, the fires burned over 8,500 hectares by February 3 [Cas24]. The death toll is at least 131 people [AP24]. Fig. 4.4 shows Landsat 8 and 9 false color imagery of two burn scars [NAS24].

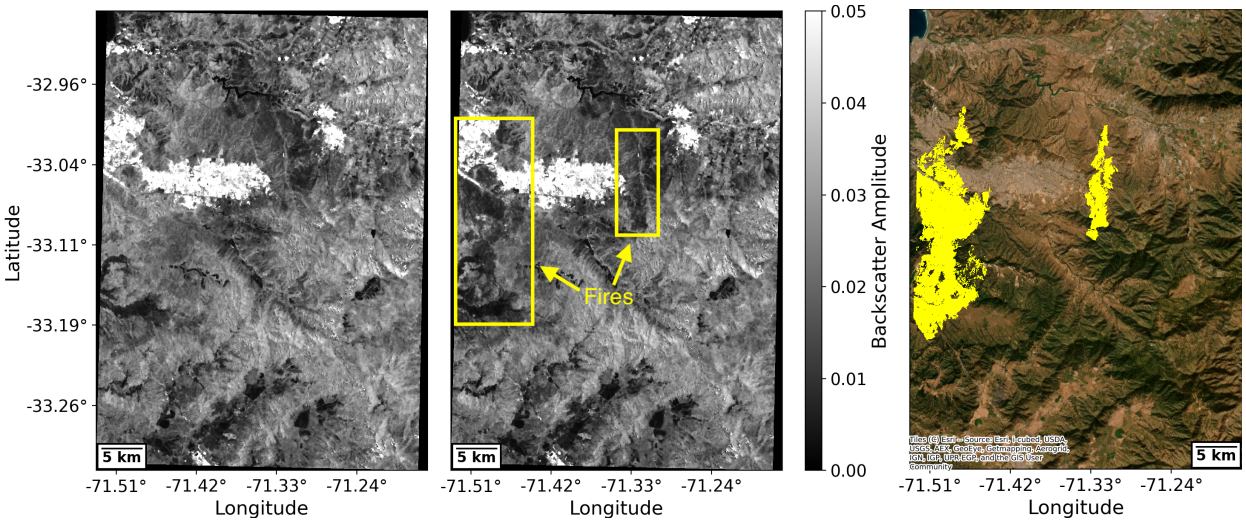


Figure 4.6: **Left**: Pre-fire SAR imagery (VH), January 23 2024. **Middle**: Post-fire SAR imagery (VH), February 28 2024. **Right**: Target (ground truth) damage map from Copernicus overlaid on ESRI World Imagery [ESR24a]. The damaged areas are shown in yellow. Note that the ESRI world imagery is a composite and thus does not capture the recent fires, and is used as a reference only. [HMH25, Figure 6]

For this site, The Copernicus Emergency Management Service (CEMS) [Cop] produced a disturbance map of the region that we use as our ground truth, see Fig. 4.6. The area of interest is approximately 45 km $\times$ 33 km. The data was generated by a variety of VHR sources and was continuously updated for several months after the fires started [ESR24b]. Our RTC-S1 data used included all seven available pre-event SAR images from 11/12/24 to 1/23/24 (all Sentinel-1 passes resulted in distributed data) and one post-event SAR image from 2/29/24. Although there was data collected and distributed on February 5 and 17 by Sentinel-1 as the fires were ongoing, there was noticeable increase in disturbance in the February 29 data. This acquisition was also closer in time to the VHR map, which was updated well beyond the fire's start [ESR24b].

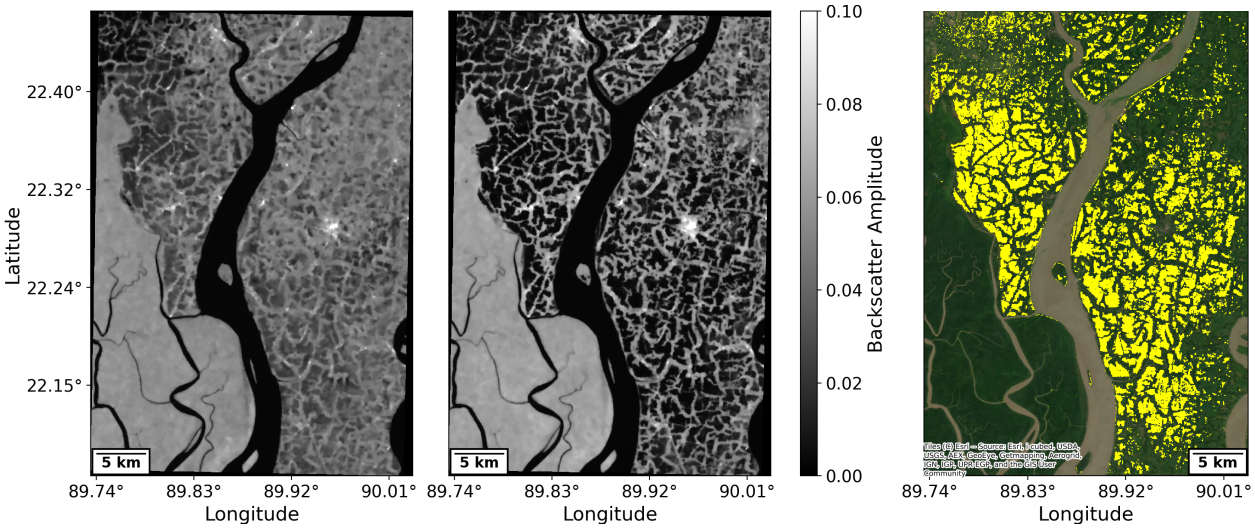


Figure 4.7: **Left**: Pre-flooding SAR imagery (VH), May 18 2024. **Middle**: Post-flooding SAR imagery (VH), July 5 2024. The flooding is seen via the reduction of the backscatter all around the Baleshwari River. **Right**: Target (ground truth) damage map from UNOSAT, overlaid on ESRI World Imagery [ESR24a]. The damaged areas are shown in yellow. Note that the ESRI imagery is a composite and thus does not capture the recent flooding, and is used as a reference only. We note that flooding continued throughout the summer, and we use data collected on July 5 as our post-event imagery to align with the damage map from UNOSAT. [HMH25, Figure 7]

4.5.3 2024 Bangladesh Floods

On May 26, 2024, Cyclone Remal landed in Bangladesh, causing major flooding that has continued throughout the summer [UNI24]. At least 4.6 million people were affected throughout the summer [UNI24]. Because the flooding occurred throughout the summer, it was challenging to establish a baseline of imagery in the RTC-S1 archive in which these floods were not present. We selected validation data containing 4 pre-event images from entirely before the cyclone, acquired between 3/31/24 and 5/18/24. We use a ground truth damage map from the United Nations Satellite Center (UNOSAT) which was derived from a Sentinel-1 image on July 4th [Hum24]. We use the nearest RTC-S1 post-event image, which was acquired

Model	Landslide PR AUC	Landslide Max F_1	Fire PR AUC	Fire Max F_1	Flood PR AUC	Flood Max F_1
Transformer	**0.732**	**0.769**	**0.680**	**0.645**	**0.754**	**0.701**
RNN	0.642	0.699	0.624	0.596	0.705	0.661
Log Ratio	0.067	0.194	0.391	0.474	0.715	0.655

Table 4.1: Summary of Results [HMH25, Table 1]

July 5th, 2024. Our flooding analysis is a subset of the country along Ganges delta around the Baleshwari River. The area of interest is approximately 42 km × 30 km, see Fig. 4.7.

4.6 Results

4.6.1 Evaluation Procedure and Performance Metrics

We evaluate and compare the performance of our proposed self-supervised transformer, the RNN from [SKZ21] adapted for use with RTC-S1, and the classical log ratio approach with per-pixel medians from the baseline set [RV93]. The definition of these metrics and how they utilize dual-polarization imagery is found in Section 4.3.2.

For our evaluation procedure, our baseline imagery is established from the dates in Section 4.5. However, we set aside the last pre-event image (i.e. the last available image before the event). The post-event image and this final pre-event image are then used to assess the disturbance delineations. We assume 1) the final pre-event image has *no* disturbance and 2) the post-event image matches the external disturbance delineation. This way, we evaluate each method on its ability to accurately delineate disturbances while also not raising false positives when we expect no disturbances. This is an important consideration in practice. In a disaster response scenario, false positives can be extremely costly; resources could be allocated to areas where they are not needed at the expense of areas that do.

Our analysis combines qualitative and quantitative results to facilitate visual comparisons and highlight the strength of our approach. Qualitative results include two map types: (1) disturbance metric maps visualizing d (transformer or RNN) or ℓ (log ratio) across the area of interest, and (2) binary (i.e., disturbance or no disturbance) disturbance delineations based on the optimal threshold τ (according to F_1 score, see below). Operationally, ground truth information is not available, so these maps are the critical outputs that delineate disturbances. In our experiments, these maps highlight the transformer's effectiveness in identifying disturbances while limiting false positives.

For our quantitative results, we report Precision-Recall (PR) curves, the associated area under the curve (AUC), and F_1 score (harmonic mean of precision and recall) [Mur12]. Precision is the percentage of pixels predicted as damage by the model that are actually damaged (positive predictive rate), while recall is the percentage of all damaged pixels that are identified (true positive rate). F_1 score is then computed by

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.6.1}$$

The ranges of all these metrics is $[0, 1]$, where higher is better. The precision-recall curves are found by varying the threshold τ (see Section 4.3.2) and computing the above metrics. A high quality but imperfect classifier will naturally trade off between precision and recall; high thresholds will result in high precision but low recall, and low thresholds will result in low precision and high recall. Hence, the area under the precision-recall curve (PR AUC) is an overall measurement of classifier quality that accounts for the entire range of τ. (A perfect classifier will achieve an AUC of 1, while a random classifier will be a horizontal line with a y-intercept equal to the percentage of positive samples in the dataset.) In all of our experiments, our transformer-based approach achieves the highest PR AUC.

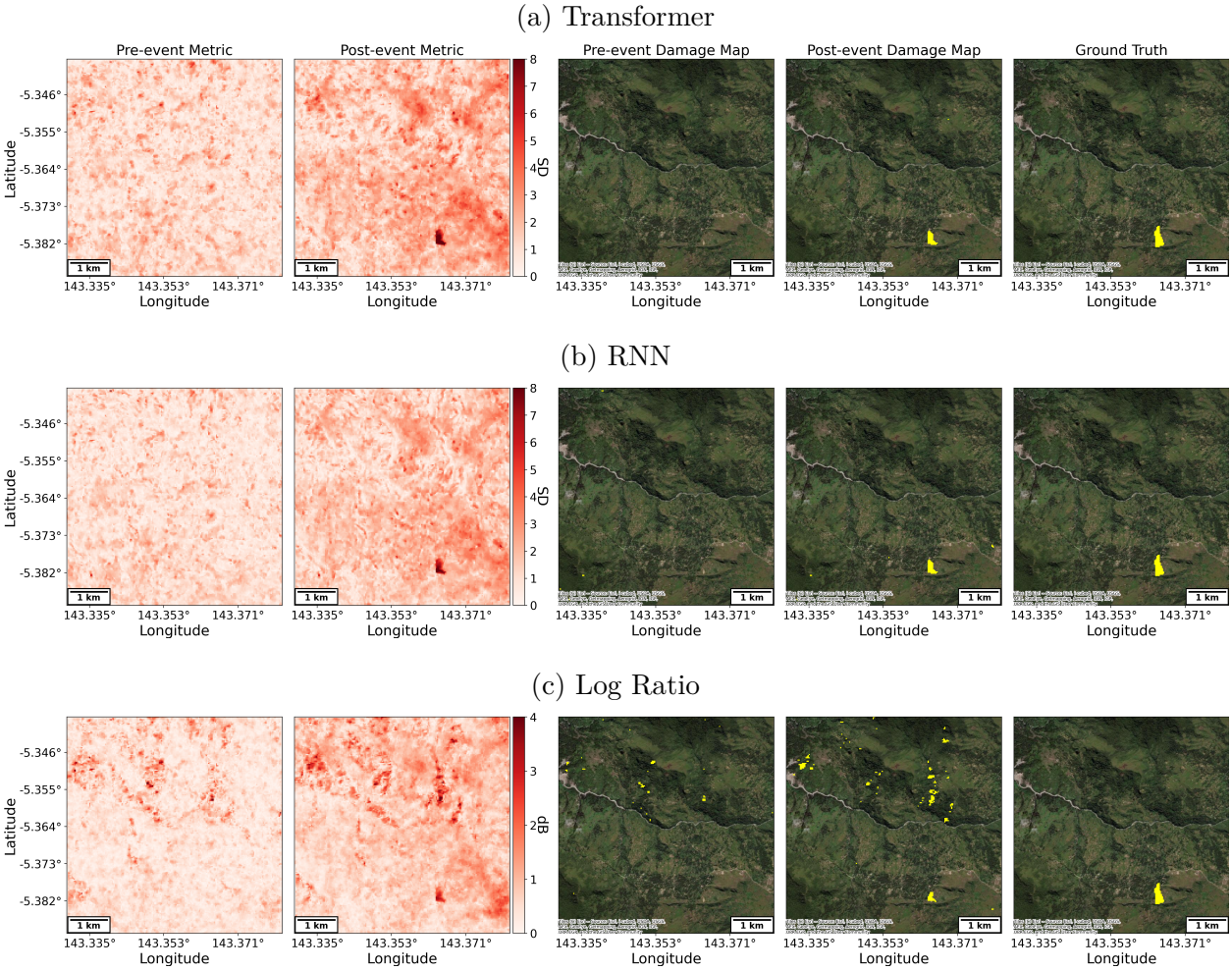


Figure 4.8: Disturbance metrics, damage maps, and ground truth data from the landslide in Papua New Guinea. The first two columns show the model's disturbance metric before the landslide (first column) and after the landslide (second column). The third and fourth columns show the best (in terms of F_1 score) binary damage maps derived from each method's metric. The rightmost column shows the ground truth damage map. The transformer metric is the strongest, resulting in the damage map with the fewest false positives while also accurately capturing the extents of the landslide. [HMH25, Figure 8]

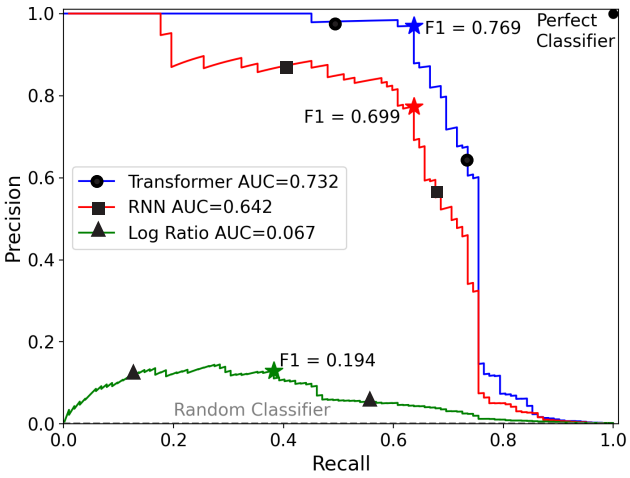


Figure 4.9: PR curves for the landslide scene. The transformer achieves the largest area under the curve (AUC). The transformer also obtains the best F_1 score (starred for each model). The disturbance delineations corresponding to the optimal τ that obtains optimal F_1 score are shown in Fig. 4.8, third and fourth columns. [HMH25, Figure 9]

Accuracy and Receiver Operating Characteristic (ROC) curves are two common metrics which we do not report here, as they are not as meaningful in datasets with significant class imbalances [DG06]; this is the case in disturbance mapping, where we typically expect large amounts of non-damaged (negative) pixels and smaller amounts of damaged (positive) pixels[6].

4.6.2 2024 Papua New Guinea Landslide

Fig. 4.8 shows the transformer metric (see Section 4.3.2), the RNN metric, and the log ratio metric. The first column is the final pre-event image and the second column is the post-event image (see Section 4.5.1). The two deep methods, particularly the transformer, are

[6]For example, in a scene with 99% negative and 1% positive pixels, a naive model which classifies every pixel as negative (and hence not delineate any disturbance extents) would achieve 99% accuracy, but 0 recall and undefined precision.

significantly more accurate than the log ratio with respect to our validation data, identifying the landslide extents more confidently and limiting the number of false positives particularly in the first pre-event image.

Fig. 4.9 shows the PR curves for the landslide scenes. The transformer achieves by far the best AUC (0.732) indicating overall model quality independent of threshold τ. The RNN is second with an AUC of 0.642, and both deep methods significantly outperform the classical log ratio method (AUC of 0.067), which barely outperforms random guessing. The optimal τ and corresponding F_1 score is starred. The transformer obtains the best F_1 score of 0.769, the RNN 0.699, and the log ratio 0.194. The disturbance delineations corresponding to the optimal τ are shown in Fig. 4.8, third and fourth columns. While in practice, the optimal τ would not be known *a priori*, we include these figures to highlight an example disturbance delineation from each model.

4.6.3 2024 Valparaíso, Chile Wildfire

Fig. 4.10 shows disturbance metrics for the transformer, RNN, and log-ratio as in the previous section. As before, the transformer is the best performing, followed by the RNN. Both deep methods are significantly more accurate than the log ratio, both in terms of identifying the fire extents confidently and limiting the number of false positives elsewhere in the scene, or before the fire occurred.

Fig. 4.11 shows the PR curves for the Chilean fire scenes. As with the Papua New Guinea landslide, the transformer achieves the best AUC of 0.680. The RNN is a second with an AUC of 0.624, and both deep methods significantly outperform the log ratio (0.391). The transformer also obtains the best F_1 score of 0.645 (the best F_1 obtained by each method is starred). The damage maps corresponding to those best F_1 scores are shown in Fig. 4.10, third and fourth columns. These binary damage maps reflect the deep learning methods' - especially the transformer's - ability to accurately map damage extents while limiting false positives.

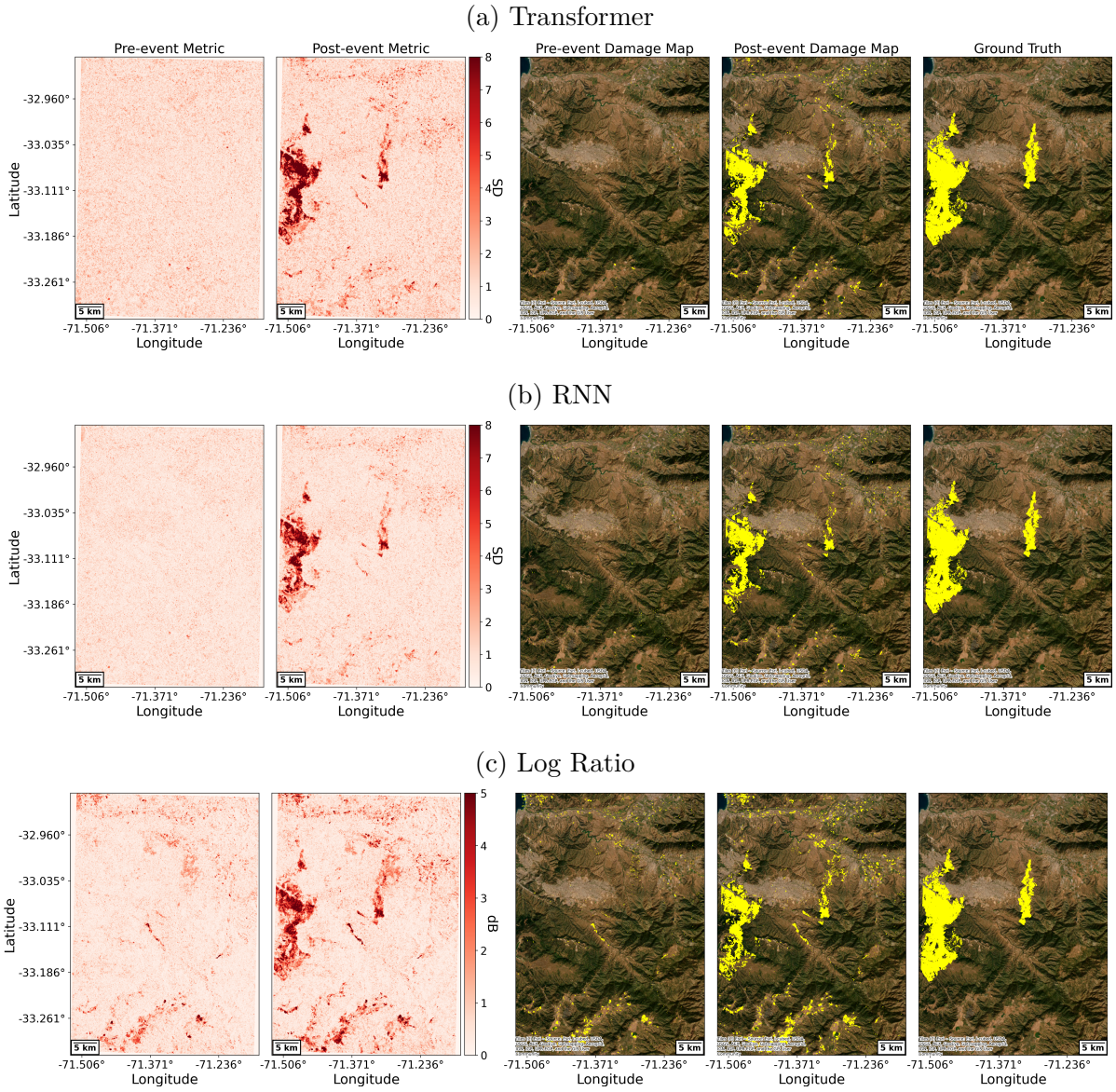


Figure 4.10: Disturbance metrics, damage maps, and ground truth data of the fire. The first two columns show the model's disturbance metric before and after the fire. The third and fourth columns show the best (by F_1 score) binary damage maps of each method. The right column shows the ground truth damage map. The transformer metric most accurately and confidently captures the fire extents while limiting false positives. [HMH25, Figure 10]

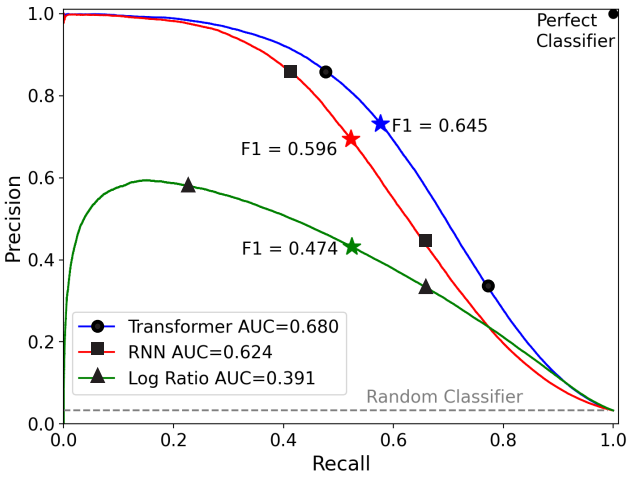


Figure 4.11: PR curves for the Chilean fire scene. The transformer is the most performant model, achieving the largest area under the curve. The transformer also obtains the best F_1 score (starred for each model). The binary damage maps corresponding to these starred F_1 scores are shown in Fig. 4.10, third and fourth columns. [HMH25, Figure 11]

4.6.4 2024 Bangladesh Floods

As in the previous two sites, we study the transformer metric, the RNN metric, and the log-ratio metric for the pre-event and post-event image in Fig. 4.12. The disturbance is more extensive in this scene and it is difficult to draw any immediate qualitative conclusions from these maps. However, the log ratio method is somewhat confident that parts of the river are a flooded area; the deep methods do not suffer as much from this (per the ground truth damage map in Fig. 4.7, the river is not considered a flooded area).

The quantitative results again show that the transformer is the most performant method. Fig. 4.13 shows the PR curves for the flood scene. The transformer achieves the best AUC 0.754. The RNN is quite similar to the classical log ratio method (AUCs of 0.705 and 0.715, respectively), underscoring the improved performance of the transformer compared to the RNN. The transformer also obtains the best F_1 score (0.701). The disturbance delineations corresponding to best F_1 scores are shown in Fig. 4.12.

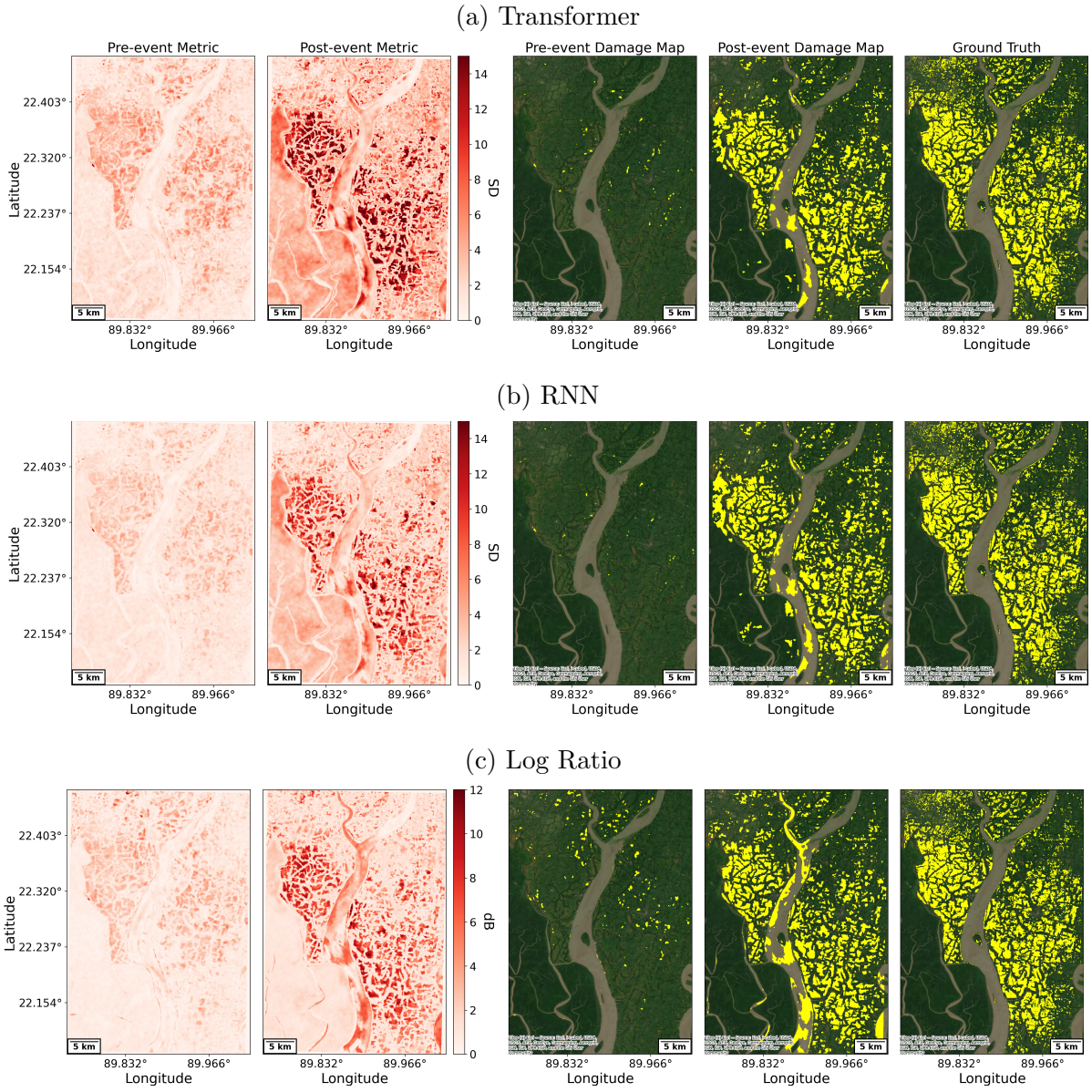


Figure 4.12: Disturbance metrics, damage maps, and ground truth data from the floods. The first two columns show the model's disturbance metric before and after the flood. The third and fourth columns show the best (by F_1 score) binary damage maps of each method. The rightmost column shows the ground truth damage map. The transformer metric is best, capturing more of the true flooding extents with fewer false positives. [HMH25, Figure 12]

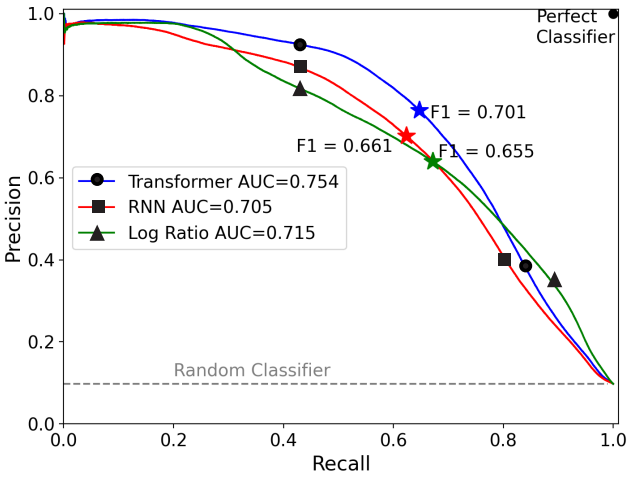


Figure 4.13: PR curves for the Bangladesh flood scene. The transformer is the most performant model, achieving the largest area under the curve. The transformer also obtains the best F_1 score (starred for each model). The binary disturbance maps corresponding to the starred F_1 scores are shown in Fig. 4.12, third and fourth columns. [HMH25, Figure 13]

4.6.5 Sensitivity to Threshold

The preceding subsections presented PR curves (where the threshold τ is varied) and best damage maps (where τ is chosen based on F_1 score relative to a ground truth). However, operationally, we typically do not have access to ground truth, so choosing τ must be done *a priori*. Hence, some discussion of the sensitivity of the methods to τ is warranted.

First, we evaluate the sensitivity of each method's F_1 score to the threshold τ for the landslide event in Fig. 4.14. Natively, the deep methods' threshold is in units of standard deviations. For the log ratio, the threshold is in units of decibels. However, to compare all the methods side-by-side, we plot the thresholds on the x-axis as a percentage of the maximum threshold (where no pixels would be labeled positive). As described previously, the transformer achieves the best F_1 score. However, this plot illuminates that the transformer is also *less sensitive to the threshold* τ compared to the RNN and log ratio at their respective peaks.

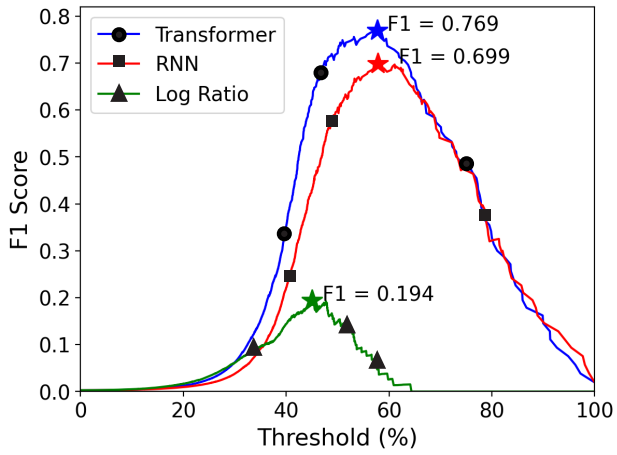


Figure 4.14: F_1 score vs. threshold for each method on the Papua New Guinea landslide. The threshold percentage is computed with respect to the maximum value (SD or dB) in the scene in order to compare all methods side-by-side. The transformer achieves the highest F_1 score, and does so with a wider peak compared to the other methods, which indicates less sensitivity to the choice of threshold τ. [HMH25, Figure 14]

Furthermore, in Fig. 4.15, we show that the transformer achieves strong F_1 scores on all tasks at similar thresholds; a threshold of approximately 5 standard deviations would result in F_1 scores around 0.6 on all three tasks. While further validation is required, this is initial evidence that one could *a priori* select $\tau \approx 5$ and expect the transformer to produce high quality disturbance delineations across environments - even when ground truth is not available. This is a critical quality for a model to possess for operational deployment. Future work will focus on further corroborating this claim on more disturbance events.

4.6.6 Ablation Studies

This subsection describes and empirically justifies some of the transformer and RNN hyperparameter choices made in this chapter. Throughout the ablation studies, we report PR AUC on each damage mapping task (landslide, fire, and flood) and bold the best result.

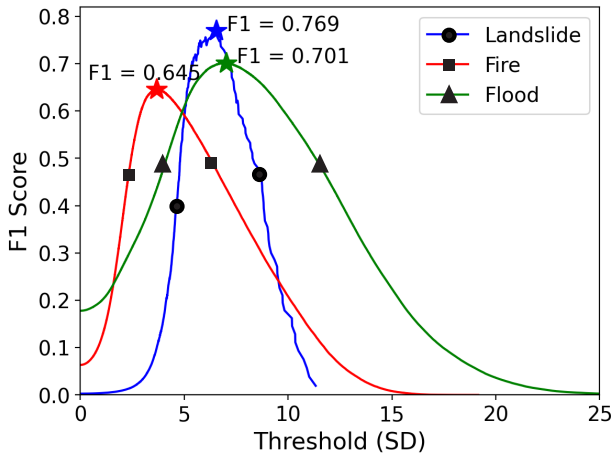


Figure 4.15: F_1 score vs thresholds for the transformer metric. Effective thresholds are consistent across all three scenes and lie in the range of 4-7 standard deviations. This indicates some generality in the model and is evidence toward operational effectiveness, where the threshold τ must be chosen without knowledge of ground truth. [HMH25, Figure 15]

4.6.6.1 Input and Patch Size

Table 4.2 compares the performance of the same transformer-based model with different input and patch sizes. The architecture and training procedure for each model in the table is the same as reported in Section 4.3.3.

I,P	Landslide PR AUC	Fire PR AUC	Flood PR AUC
16,8	**0.732**	**0.680**	0.754
32,8	0.714	0.630	**0.771**
32,16	0.630	0.603	0.744

Table 4.2: Effect of input (I) and patch sizes (P) on transformer performance. [HMH25, Table 2]

While vision transformers for image classification tasks typically use 224×224 as the

Input Size	Landslide PR AUC	Fire PR AUC	Flood PR AUC
8×8	**0.642**	**0.624**	**0.705**
16×16	0.334	0.540	0.661

Table 4.3: Effect of input size on RNN performance. [HMH25, Table 3]

input size with a 16×16 patch size [DBK20], we found that a smaller input size (16) and - in particular - a smaller patch size (8) was most performant on disturbance mapping tasks. One explanation of this difference is the resolution of the data; OPERA RTC-S1 data has a resolution of 30 meters, representing a completely different scale than typical (higher resolution) image data for image classification tasks. A smaller patch size provides more spatial detail in this remote sensing setting, which has been noted in previous work [CKM22]. This is especially useful in disturbance mapping, where the damage extents often have a fine-grained boundary at a 30 meter resolution.

We also experiment with the input size for the RNN in Table 4.3. Transformers applied to vision tasks process the input by patchifying and applying a positional encoding. Conversely, RNNs do not have a positional encoding. Since the optimal transformer in our application uses a 16×16 input size and an 8×8 patch size, we compare RNNs trained and deployed with a 16×16 and an 8×8 input size. Similar to our findings with the transformer, the smaller input size obtains better PR AUC, particularly on the landslide scene where the damaged area is smaller.

4.6.6.2 Model Size

Table 4.4 compares the performance of the transformer-based model with varying architecture choices (and hence, varying number of total trainable parameters). The other modeling and training choices are the same as reported in Section 4.3.3, except the largest model required an initial learning rate of 1e-5 instead of 1e-4 (otherwise, the loss exploded). The model with 3.3 million parameters performs the best on each disturbance mapping task,

which is the model we used to report results in the previous sections. However, we expect that with significantly more training data, a larger model can improve performance over what is reported in this chapter. For now, we offer the comparison in Table 4.4 to justify our choices in this chapter, and leave scaling up the training set size and model size to future work.

FF, L, P	Landslide PRAUC	Fire PRAUC	Flood PRAUC
512, 2, 1.5M	0.717	0.670	0.748
768, 4, 3.3M	**0.732**	**0.680**	**0.754**
1024, 8, 7.1M	0.728	0.658	0.744

Table 4.4: Effect of width of feedforward layers (FF), number of layers (L), and total number of parameters (P) on transformer performance. [HMH25, Table 4]

4.6.6.3 Learning Rate

Finally, we ablate the learning rate. Table 4.5 compares the transformer model trained at different starting learning rates (with the same learning rate decay factor of 0.1 at epoch 25). The other modeling and training choices are the same as reported in Section 4.3.3. The starting learning rate of 1e-4 exhibits the strongest results, which corresponds to the results we report in the main text. Learning rates larger than 1e-4 resulted in little to no training and are omitted.

LR	Landslide PR AUC	Fire PR AUC	Flood PR AUC
1e-4	**0.732**	**0.680**	**0.754**
1e-5	0.725	0.632	0.746
1e-6	0.648	0.551	0.678

Table 4.5: Effect of learning rate on transformer performance. [HMH25, Table 5]

4.7 Discussion

The preceding subsections demonstrate the strength of our transformer-based disturbance algorithm on three natural disasters, outperforming the RNN and log ratio on all three sites as measured by area under the precision-recall curve. Our results suggest that the data-driven machine learning methods, especially the transformer, are better disturbance quantification tools than the classical log-ratio approach, provided one has access to sufficient high quality training data collected for self-supervised training. Indeed, a single, relatively small transformer was able to outperform the log ratio test significantly and consistently across experiments. Our results also illustrate the improved performance and generality of our work compared to previous deep learning-based approaches [SKZ21]. Overall, our results demonstrate the effectiveness of our self-supervised transformer across different environments and disasters.

Although we are far from verifying this approach is globally applicable, the transformer was consistently the best performing method on three different disaster events in three different parts of the world. We also note that the optimal thresholds for the transformer all fall within 4 to 7 standard deviations (see Fig. 4.15); in particular, $\tau \approx 5$ would produce $F_1 \approx 0.6$ across all three sites. This indicates some inherent generality in the model, especially for operational purposes, where the threshold τ would have to be chosen without knowledge of ground truth. In practice, the choice of threshold will depend on the application: a lower threshold will capture more true positives (that is, disturbances) at the expense of possibly more false positives, while a higher threshold will ensure fewer false positives but risk missing some true positives.

Finally, a limitation of this chapter is that we evaluated methods' disturbance delineation on SAR data by comparing to disturbance maps generated using optical data. There will inherently be differences and imperfections on both optical and SAR data and the delineations derived from them. Some disturbances will be less detectable (or even invisible) in

one modality compared to the other. Moreover, there may be other disturbances in these scenes that are not accounted for in our ground truth damage maps, which only consider the disturbances caused by the disaster. There could be real changes in backscatter that are not related to the damage event, which would not be accounted for in our analysis. However, these limitations are difficult to negate, and due to the abruptness and scale of natural disasters, we expect the effect of non-disaster disturbances to be small. Moreover, the consistently high performance of the transformer indicates potential as a global disturbance model. A more complete global analysis is needed to ascertain if this transformer metric and global thresholds can be used to ascertain disturbances in a variety of environments, and be deployed for (near-)global disturbance mapping.

4.8 Conclusion and Future Work

In this chapter, we trained deep self-supervised vision transformer on near-global OPERA RTC-S1 SAR data. We systematically analyzed this transformer-based metric to show it had strong agreement with externally validated data. While many previous works on SAR disturbance require supervision or are trained on a particular location and disaster event, our model was self-supervised without needing any fine-tuning for different environments or different categories of disaster.

There are two main avenues of future work. One is to provide systematic verification that our vision transformer approach has global applicability for the RTC-S1 data. This analysis includes assessing additional categories of disturbances such as logging, urbanization, and mining; analyzing the necessary baseline imagery required to delineate disturbances; and analyzing how disturbance delineations change and can be tracked over time. The goal of such systematic analysis is to adapt this transformer disturbance algorithm for near-global disturbance monitoring through RTC-S1 inputs.

Another avenue of future work is to improve the model itself including increasing the

model's depth, training the model on more data, adding more nuanced spatiotemporal features [RGL23, TCZ23], a masked pretraining step [HCX22], and increasing the window size so that the vision transformer can expand its spatial context. The ultimate goal of this type of work is to generate a SAR foundation model that can be fine-tuned across various wavelengths (such as C-band and L-band for the coming NISAR mission) and across multiple downstream tasks including disturbance mapping, land cover use change, classification. Indeed, promising work has been done in this direction for multispectral data [Cla24, JRP23, SLC24] and hybrid remote sensing data [TCZ23, RAK20a]. We believe the community would benefit from a SAR-specific approach, especially using the new OPERA RTC-S1 product and the forthcoming NISAR products.

We hope this chapter is a step toward developing a disturbance tool that can be used to inform decision makers and emergency responders in the wake of natural disasters, and be a tool for researchers who study disturbances.

CHAPTER 5

Conclusion

This dissertation developed novel methods for label-efficient machine learning. In the semi-supervised setting, we proposed algorithms to address two major components of GBSSL: constructing a high-quality graph (Chapter 2), and leveraging the graph's structure for the learning task (Chapter 3). Striving toward full label independence, Chapter 4 introduced a self-supervised model for disturbance mapping at a near-global scale.

In Chapter 2, we described a methodology that supplies *structure from learning* for GBSSL. We computed backpropagation equations for similarity graph construction and a general family of graph learning methods in a gradient-based learning framework. Whereas previous work approximated gradients or decoupled graph learning from neural network-based encoders, our method updates parameters directly based on loss signals derived from graph learning outputs. Using this insight, we designed the Graph Learning Layer (GLL), which replaces a projection head for classification and can be implemented efficiently with minimal computational overhead. In experiments, models with GLL heads achieved superior clean and adversarial accuracy across datasets, architecture backbones, and labeling rates, with the gap increasing as the label rate shrank. We further validated the utility of exact backpropagation equations by comparing to WNLL [WO21] - which relies on gradient heuristics - where GLL exhibited consistent gains. Future work should focus on expanding experiments to other graph learning methods, such as Poisson learning [CCT20], which is designed for especially low label rates. Applying active learning to GLL-enabled deep neural networks is also an appealing avenue for future work.

Chapter 3 studied how to *learn from structure* in graph-based active learning. Specifically, we proposed a BFC-based coreset algorithm, Curvature Coreset (CC), that utilizes the cluster structure of the graph to build a topologically-aware coreset. CC can detect when the graph is sufficiently explored according to BFC, providing a natural, label-efficient stopping condition. As a further extension, we illustrated how BFC can improve existing AL routines, providing greater flexibility across datasets. On a variety of image datasets, we demonstrated stronger semi-supervised performance with lower computational cost in both sequential and batch AL experiments compared to baseline methods. The second thrust of this chapter *modified* the graph to incorporate higher-order interactions around labeled nodes. By localizing this multiscale regularization, we strike a desirable balance between accuracy and efficiency: our method outperforms Laplace learning while achieving a tenfold speed-up compared to the "full" multiscale method. Taken as a whole, this chapter positions graph-topological techniques as compelling tools to address the central tension between exploration and exploitation in AL, ultimately enabling stronger models with fewer labels. There are several potential directions for future work. One is to investigate how a graph's homophily (or lack thereof) influences the balance of exploration and exploitation - perhaps delineating "regimes" in which more or less exploration is needed. This chapter also highlights the utility of tools from the GNN community for GBSSL. These two communities have developed ideas mostly in isolation from each other, despite studying the same underlying mathematical structures. The cross-pollination of ideas between GNN and GBSSL is indeed a fertile ground for future efforts.

Finally, in Chapter 4, we developed a deep vision transformer for near-global disturbance monitoring using recently-released RTC SAR data from Sentinel-1. We employed self-supervised training to eliminate dependence on time-consuming labeled data sets and ensure scalability given huge amounts of unlabeled data, and tailored the outputs to provide an interpretable measure of confidence in disturbance delineation via the Mahalanobis distance. In contrast to previous works, our model is highly general and can be applied

to varying disturbances and terrains without retraining or supervision. We validated the model on several disturbance mapping tasks from recent real-world natural disasters, where we improved upon prior methods by up to an order of magnitude in PR AUC. Future work should focus on larger-scale validation of the model, such as evaluating on more events and types of disturbance (urbanization, logging, mining). The model itself could also likely be improved by increasing the number of parameters, amount of training data, and the context window, in line with recent trends in deep learning. With the recent launch of NISAR, the availability of SAR data and the potential for high-impact applications will continue to grow rapidly. The long-term vision of this line of work is to produce a SAR foundation model that can be deployed in practice, serving stakeholders before, during, and after natural disasters.

Taken together, the dissertation elucidates a common thread: by reducing or eliminating the need for labeled data, we can unlock the potential of modern machine learning techniques for applications where supervision is costly or entirely unavailable.

REFERENCES

[AAB19] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. "Differentiable convex optimization layers." *Advances in neural information processing systems*, **32**, 2019.

[ABC24] ABC News. "First Look at the PNG Landslide Site." `https://web.archive.org/web/20240601132827/https://www.abc.net.au/news/2024-05-31/first-look-at-the-png-landslide-site/103920178`, 2024. [Archived; accessed 16-Oct-2024].

[ADG23] Matt Allen, Francisco Dorr, Joseph A Gallego-Mejia, Laura Martínez-Ferrer, Anna Jungbluth, Freddie Kalaitzis, and Raúl Ramos-Pollán. "Large Scale Masked Autoencoding for Reducing Label Requirements on SAR Data." *arXiv preprint arXiv:2310.00826*, 2023.

[AKK20] Ram Avtar, Akinola Adesuji Komolafe, Asma Kouser, Deepak Singh, Ali P Yunus, Jie Dou, Pankaj Kumar, Rajarshi Das Gupta, Brian Alan Johnson, Huynh Vuong Thu Minh, et al. "Assessing sustainable development prospects through remote sensing: A review." *Remote sensing applications: Society and environment*, **20**:100402, 2020.

[AKS21] Pukar Amatya, Dalia Kirschbaum, Thomas Stanley, and Hakan Tanyas. "Landslide mapping using object-based image analysis and open source tools." *Engineering geology*, **282**:106000, 2021.

[Ala24] Alaska Satellite Facility (ASF). "Sentinel-1 Acquisition Maps." `https://asf.alaska.edu/daac/sentinel-1-acquisition-maps/`, 2024. [Online; accessed 16-Oct-2024].

[AP24] AP. "The death toll from Chile's wildfires reaches 131, and more than 300 people are missing." `https://apnews.com/article/chile-deadly-wildfires-valparaiso-d40a40ed31d14d7901080f0d700d2afa`, 2024.

[ASK22] Yusupujiang Aimaiti, Christina Sanon, Magaly Koch, Laurie G Baise, and Babak Moaveni. "War related building damage assessment in Kyiv, Ukraine, using Sentinel-1 radar and Sentinel-2 optical images." *Remote Sensing*, **14**(24):6239, 2022.

[BBC23] M. Grace Bato, David Bekaert, Bruce Chapman, Matt Hansen, John W. Jones, Jin Woo Kim, and Others. "OPERA Product Validation Document v1.5." `https://t.ly/CrUq`, 2023.

[BBM05] Y. Bazi, L. Bruzzone, and F. Melgani. "An Unsupervised Approach based on the Generalized Gaussian Model to Automatic Change Detection in Multitemporal SAR images." *IEEE Transactions on Geoscience and Remote Sensing*, **43**(4):874–887, 2005.

[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.

[BCH24] Jason Brown, Bohan Chen, Harris Hardiman-Mostow, Jeff Calder, and Andrea L Bertozzi. "GLL: A Differentiable Graph Learning Layer for Neural Networks." *arXiv preprint arXiv:2412.08016*, 2024.

[BDL06] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. *Label Propagation and Quadratic Criterion*, pp. 193–216. MIT Press, semi-supervised learning edition, January 2006.

[Ber24] Erik Bernhardsson. "ANNOY library." `https://github.com/spotify/annoy`, 2024. Accessed: 2024-01-01.

[BF16] Andrea L Bertozzi and Arjuna Flenner. "Diffuse interface models on graphs for classification of high dimensional data." *SIAM Review*, **58**(2):293–328, 2016.

[BLS18] Andrea L. Bertozzi, Xiyang Luo, Andrew M. Stuart, and Konstantinos C. Zygalakis. "Uncertainty quantification in graph-based classification of high-dimensional data." *SIAM/ASA Journal on Uncertainty Quantification*, **6**(2):568–595, 2018.

[BMN04] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. "Regularization and semi-supervised learning on large graphs." In *Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004. Proceedings 17*, pp. 624–638. Springer, 2004.

[BMR20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever,

and Dario Amodei. "Language Models are Few-Shot Learners." In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

[BN04] Mikhail Belkin and Partha Niyogi. "Semi-supervised learning on Riemannian manifolds." *Machine learning*, **56**:209–239, 2004.

[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.

[BP00] L. Bruzzone and D.F. Prieto. "Automatic Analysis of the Difference Image for Unsupervised Change Detection." *IEEE Transactions on Geoscience and Remote Sensing*, **38**(3):1171–1182, 2000.

[BS91] Guy Barles and Panagiotis E Souganidis. "Convergence of approximation schemes for fully nonlinear second order equations." *Asymptotic analysis*, **4**(3):271–283, 1991.

[BSB19] Luca Bergamasco, Sudipan Saha, Francesca Bovolo, and Lorenzo Bruzzone. "Unsupervised change-detection based on convolutional-autoencoder feature extraction." In *Image and Signal Processing for Remote Sensing XXV*, volume 11155, pp. 325–332. SPIE, 2019.

[BTA20] Derrick Bonafilia, Beth Tellman, Tyler Anderson, and Erica Issenberg. "Sen1Floods11: a georeferenced dataset to train and test deep learning flood algorithms for Sentinel-1." In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 835–845, 2020.

[BW20] Juliet Biggs and Tim J Wright. "How satellite InSAR has grown from opportunistic science to routine monitoring over the last decade." *Nature Communications*, **11**(1):3863, 2020.

[BZN20] Yifang Ban, Puzhao Zhang, Andrea Nascetti, Alexandre R Bevington, and Michael A Wulder. "Near real-time wildfire progression monitoring with Sentinel-1 SAR time series and deep learning." *Scientific reports*, **10**(1):1322, 2020.

[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 J. on Mathematics of Data Science*, **1**(4):780–812, January 2019.

[Cal22] Jeff Calder. "GraphLearning Python Package.", January 2022.

[Can19] Morton John Canty. *Image analysis, classification and change detection in remote sensing: with algorithms for Python*. Crc Press, 2019.

[Cas24] Emily Cassidy. "Fires Rage in Central Chile." `https://earthobservatory.nasa.gov/images/152411/fires-rage-in-central-chile`, 2024.

[CAT17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. "EMNIST: An extension of MNIST to handwritten letters.", 2017.

[CCG24] Daniele Rege Cambrin, Isaac Corley, and Paolo Garza. "Depth any canopy: Leveraging depth foundation models for canopy height estimation." *arXiv preprint arXiv:2408.04523*, 2024.

[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 to synthetic aperture radar datasets." In *Algorithms for Synthetic Aperture Radar Imagery XXX*, volume 12520, pp. 95–110. SPIE, 2023.

[CD24] Jeff Calder and Nadejda Drenska. "Consistency of semi-supervised learning, stochastic tug-of-war games, and the p-Laplacian." *To appear in Active Particles, Volume 4, Advances in Theory, Models, and Applications*, 2024.

[CE22] Jeff Calder and Mahmood Ettehad. "Hamilton-Jacobi equations on graphs with applications to semi-supervised learning and data depth." *Journal of Machine Learning Research*, **23**(318):1–62, 2022.

[CG22] Jeff Calder and N. García Trillos. "Improved spectral convergence rates for graph Laplacians on ε-graphs and k-NN graphs." *Applied and Computational Harmonic Analysis*, **60**:123–175, 2022.

[Cho14] Kyunghyun Cho. "On the Properties of Neural Machine Translation: Encoder-decoder Approaches." *arXiv preprint arXiv:1409.1259*, 2014.

[CKM22] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. "Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery." *Advances in Neural Information Processing Systems*, **35**:197–211, 2022.

[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.

[Cla24] Clay Foundation and Development Seed LLC. "Model Repository." https://github.com/Clay-foundation/model, 2024. Accessed: 2024-10-11.

[CM21] Alexander Cloninger and Hrushikesh N Mhaskar. "Cautious active clustering." *Applied and Computational Harmonic Analysis*, **54**:44–74, 2021.

[CMB24] 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*, **6**(2):1013–1033, 2024.

[CMB25] 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*, **18**:446–462, 2025.

[Cop] Copernicus Emergency Management System © 2024 European Union, EMSR715. `https://rapidmapping.emergency.copernicus.eu/EMSR715/download`.

[Cop24] Copernicus Emergency Management Service. "Copernicus EMS Rapid Mapping." `https://rapidmapping.emergency.copernicus.eu/`, 2024. [Online; accessed 16-Oct-2024].

[CRO24] Isaac Corley, Caleb Robinson, and Anthony Ortiz. "A Change Detection Reality Check." *arXiv preprint arXiv:2402.06994*, 2024.

[CS20a] Jeff Calder and Dejan Slepcev. "Properly-weighted graph Laplacian for semi-supervised learning." *Applied Mathematics & Optimization*, **82**:1111–1159, 2020.

[CS20b] Hao Chen and Zhenwei Shi. "A Spatial-Temporal Attention-Based Method and a New Dataset for Remote Sensing Image Change Detection." *Remote Sensing*, **12**(10), 2020.

[CSH18] Philip G Curtis, Christy M Slay, Nancy L Harris, Alexandra Tyukavina, and Matthew C Hansen. "Classifying drivers of global forest loss." *Science*, **361**(6407):1108–1111, 2018.

[CSH24] Hongruixuan Chen, Jian Song, Chengxi Han, Junshi Xia, and Naoto Yokoya. "Changemamba: Remote sensing change detection with spatio-temporal state space model." *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[CST23] Jeff Calder, Dejan Slepcev, and Matthew Thorpe. "Rates of convergence for Laplacian semi-supervised learning with low labeling rates." *Research in the Mathematical Sciences*, **10**(1):10, 2023.

[CW17] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks." In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.

[CWD24] Chaoqi Chen, Yushuang Wu, Qiyuan Dai, Hong-Yu Zhou, Mutian Xu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. "A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[CZS19] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. "RandAugment: Practical data augmentation with no separate search." *CoRR*, **abs/1909.13719**, 2019.

[DBK20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In *International Conference on Learning Representations*, 2020.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems*, **29**, 2016.

[DDT12] Charles-Alban Deledalle, Loïc Denis, and Florence Tupin. "How to Compare Noisy Patches? Patch Similarity beyond Gaussian Noise." *International journal of computer vision*, **99**:86–102, 2012.

[DDT17] Charles-Alban Deledalle, Loic Denis, Sonia Tabti, and Florence Tupin. "MuLoG, or how to apply Gaussian denoisers to multi-channel SAR speckle reduction?" *IEEE Transactions on Image Processing*, **26**(9):4389–4403, 2017.

[Dev18] Jacob Devlin. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805*, 2018.

[DG06] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

[DHA20] Ben DeVries, Chengquan Huang, John Armston, Wenli Huang, John W Jones, and Megan W Lang. "Rapid and robust monitoring of flood events using Sentinel-1 and Landsat data on the Google Earth Engine." *Remote Sensing of Environment*, **240**:111664, 2020.

[DSS20] Matthew M Dunlop, Dejan Slepcev, Andrew M Stuart, and Matthew Thorpe. "Large data and zero noise limits of graph-based semi-supervised learning algorithms." *Applied and Computational Harmonic Analysis*, **49**(2):655–697, 2020.

[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.

[EHC23] Joshua Enwright, Harris Hardiman-Mostow, Jeff Calder, and Andrea Bertozzi. "Deep semi-supervised label propagation for SAR image classification." In *Algorithms for Synthetic Aperture Radar Imagery XXX*, volume 12520, pp. 158–170. SPIE, 2023.

[ESR24a] ESRI from DigitalGlobe, GeoEye, iCubed, USDA FSA, USGS, AEX, Getmapping, Aerogrid, IGN, IGP, swisstopo, and the GIS User Community. "Esri World Imagery." `https://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9`, 2024. [Online; accessed 16-Oct-2024].

[ESR24b] ESRI StoryMaps. "Inferno Scars Valparaíso - StoryMaps." `https://storymaps.arcgis.com/stories/6182cb43b76e4aaab634c4b5eb1b423a`, 2024. [Online; accessed 16-Oct-2024].

[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.

[FFL22] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. "Masked autoencoders as spatiotemporal learners." *Advances in neural information processing systems*, **35**:35946–35958, 2022.

[FW24] Amy Feng and Melanie Weber. "Graph Pooling via Ricci Flow." *Transactions on Machine Learning Research*, 2024.

[GGH20] Nicolás García Trillos, Moritz Gerlach, Matthias Hein, and Dejan Slepcev. "Error estimates for spectral convergence of the graph Laplacian on random geometric graphs toward the Laplace–Beltrami operator." *Foundations of Computational Mathematics*, **20**(4):827–887, 2020.

[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.

[GS18] Nicolas Garcia Trillos and Dejan Slepcev. "A variational approach to the consistency of spectral clustering." *Applied and Computational Harmonic Analysis*, **45**(2):239–281, 2018.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572*, 2014.

[HAL05] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. "From Graphs to Manifolds-Weak and Strong Pointwise Consistency of Graph Laplacians." In *COLT*, volume 3559, pp. 470–485. Springer, 2005.

[HAL07] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. "Graph Laplacians and their convergence on random neighborhood graphs." *Journal of Machine Learning Research*, **8**(6), 2007.

[HCX22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked autoencoders are scalable vision learners." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

[HHJ22] Alexander L Handwerger, Mong-Han Huang, Shannan Y Jones, Pukar Amatya, Hannah R Kerner, and Dalia B Kirschbaum. "Generating landslide density heatmaps for rapid detection using open-access satellite radar data in Google Earth Engine." *Natural Hazards and Earth System Sciences*, **22**(3):753–773, 2022.

[HHO22] Franca Hoffmann, Bamdad Hosseini, Assad A. Oberai, and Andrew M. Stuart. "Spectral analysis of weighted Laplacians arising in data clustering." *Applied and Computational Harmonic Analysis*, **56**:189–249, 2022.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in neural information processing systems*, **33**:6840–6851, 2020.

[HLS05] Jihun Ham, Daniel Lee, and Lawrence Saul. "Semisupervised alignment of manifolds." In *International Workshop on Artificial Intelligence and Statistics*, pp. 120–127. PMLR, 2005.

[HMH25] Harris Hardiman-Mostow, Charles Marshak, and Alexander L Handwerger. "Deep Self-Supervised Disturbance Mapping with the OPERA Sentinel-1 Radiometric Terrain Corrected SAR Backscatter Product." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2025.

[HMW25] Harris Hardiman-Mostow, Jack Mauro, Adrien Weihs, and Andrea L Bertozzi. "Topology-Aware Active Learning on Graphs." *arXiv preprint arXiv:2510.25892*, 2025.

[HPM13] Matthew C Hansen, Peter V Potapov, Rebecca Moore, Matt Hancher, Svetlana A Turubanova, Alexandra Tyukavina, David Thau, Stephen V Stehman, Scott J Goetz, Thomas R Loveland, et al. "High-resolution global maps of 21st-century forest cover change." *science*, **342**(6160):850–853, 2013.

[HPS24] Matt Hansen, Amy Pickens, and Zhen Song. "OPERA DIST Product - Algorithm Theoretical Basis Document." `https://lpdaac.usgs.gov/documents/1835/OPERA_DIST_ATBD_V1.pdf`, 2024.

[HSB15] Huiyi Hu, Justin Sunu, and Andrea L. Bertozzi. "Multi-class Graph Mumford-Shah Model for Plume Detection using the MBO scheme." In *Springer Lecture Notes in Computer Science*, volume 8932, pp. 209–222, 2015.

[Hum24] Humanitarian Data Exchange (HDX). "Satellite-detected Water Extents Between 4-11 July 2024 in Bangladesh." `https://data.humdata.org/dataset/satellite-detected-water-extents-between-4-11-july-2024-in-bangladesh`, 2024. [Online; accessed 16-Oct-2024].

[HWG22] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. "Vision gnn: An image is worth graph of nodes." *Advances in neural information processing systems*, **35**:8291–8303, 2022.

[HZG23] Zhu Han, Ce Zhang, Lianru Gao, Zhiqiang Zeng, Bing Zhang, and Peter M Atkinson. "Spatio-temporal multi-level attention crop mapping method using time-series SAR imagery." *ISPRS Journal of Photogrammetry and Remote Sensing*, **206**:293–310, 2023.

[HZR16a] 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.

[HZR16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016.

[HZZ21] Huaxi Huang, Junjie Zhang, Jian Zhang, Qiang Wu, and Chang Xu. "PTN: A Poisson transfer network for semi-supervised few-shot learning." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1602–1609, 2021.

[IBM24] IBM and NASA. "HLS Burn Scars Dataset – Hugging Face." `https://huggingface.co/datasets/ibm-nasa-geospatial/hls_burn_scars`, 2024. [Online; accessed 16-Oct-2024].

[ICB20] 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, 2020.

[ITA19] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Label propagation for deep semi-supervised learning." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5070–5079, 2019.

[JH12] Ming Ji and Jiawei Han. "A Variance Minimization Criterion to Active Learning on Graphs." In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 556–564, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[JKL16] Jungkyo Jung, Duk-jin Kim, Marco Lavalle, and Sang-Ho Yun. "Coherent Change Detection Using InSAR Temporal Decorrelation Model: A Case Study for Volcanic Ash Detection." *IEEE Transactions on Geoscience and Remote Sensing*, **54**(10):5765–5775, 2016.

[Joa99] Thorsten Joachims et al. "Transductive inference for text classification using support vector machines." In *Icml*, volume 99, pp. 200–209, 1999.

[JRP23] Johannes Jakubik, Sujit Roy, CE Phillips, Paolo Fraccaro, Denys Godwin, Bianca Zadrozny, Daniela Szwarcman, Carlos Gomes, Gabby Nyirjesy, Blair Edwards, et al. "Foundation Models for Generalist Geospatial Artificial Intelligence." *arXiv preprint arXiv:2310.18660*, 2023.

[JWL19] Shunping Ji, Shiqing Wei, and Meng Lu. "Fully Convolutional Networks for Multisource Building Extraction From an Open Aerial and Satellite Imagery Data Set." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(1):574–586, 2019.

[JY20] Jungkyo Jung and Sang-Ho Yun. "Evaluation of coherent and incoherent landslide detection methods based on synthetic aperture radar for rapid response: A case study for the 2018 Hokkaido landslides." *Remote Sensing*, **12**(2):265, 2020.

[JYK17] Jungkyo Jung, Sang-Ho Yun, Duk-jin Kim, and Marco Lavalle. "Damage-mapping algorithm based on coherence model using multitemporal polarimetric–interferometric SAR data." *IEEE Transactions on Geoscience and Remote Sensing*, **56**(3):1520–1532, 2017.

[KAM24] Rıdvan Salih Kuzu, Oleg Antropov, Matthieu Molinier, Corneliu Octavian Dumitru, Sudipan Saha, and Xiao Xiang Zhu. "Forest Disturbance Detection via Self-Supervised and Transfer Learning With Sentinel-1 2 Images." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **17**:4751–4767, 2024.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KBM23] Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs." In *The Eleventh International Conference on Learning Representations*, 2023.

[KGB18] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

[KH09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." *Toronto, ON, Canada*, 2009.

[KH13] Jasper Knight and Stephan Harrison. "The impacts of climate change on terrestrial Earth surface systems." *Nature Climate Change*, **3**(1):24–29, 2013.

[KMR23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. "Segment anything." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

[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:1312.6114*, 2013.

[KW16] Thomas N Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In *International Conference on Learning Representations*, 2016.

[KWH19] Christian N. Koyama, Manabu Watanabe, Masato Hayashi, Takashi Ogawa, and Masanobu Shimada. "Mapping the spatial-temporal variability of tropical forests by ALOS-2 L-band SAR big data analysis." *Remote Sensing of Environment*, **233**:111372, 2019.

[LH16] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983*, 2016.

[LHW13] Wen-Yu Lee, Liang-Chi Hsieh, Guan-Long Wu, and Winston Hsu. "Graph-based semi-supervised learning with multi-modality propagation for large-scale image datasets." *Journal of visual communication and image representation*, **24**(3):295–302, 2013.

[LHW22] Chuxuan Li, Alexander L Handwerger, Jiali Wang, Wei Yu, Xiang Li, Noah J Finnegan, Yingying Xie, Giuseppe Buscarnera, and Daniel E Horton. "Augmentation of WRF-Hydro to simulate overland-flow-and streamflow-generated debris flow susceptibility in burn scars." *Natural Hazards and Earth System Sciences*, **22**(7):2317–2345, 2022.

[LKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization." *ArXiv e-prints*, pp. arXiv–1607, 2016.

[LLP19] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. "Learning to propagate labels: Transductive propagation network for few-shot learning." In *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR, 2019.

[LLR24] Alexandre Lacoste, Nils Lehmann, Pau Rodriguez, Evan Sherwin, Hannah Kerner, Björn Lütjens, Jeremy Irvin, David Dao, Hamed Alemohammad, Alexandre Drouin, et al. "Geo-bench: Toward foundation models for earth monitoring." *Advances in Neural Information Processing Systems*, **36**, 2024.

[LLU16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. "Understanding the effective receptive field in deep convolutional neural networks." *Advances in neural information processing systems*, **29**, 2016.

[LM12] Isaac J Larsen and David R Montgomery. "Landslide erosion coupled to tectonics and river incision." *Nature Geoscience*, **5**(7):468–473, 2012.

[Lux07] Ulrike von Luxburg. "A tutorial on spectral clustering." *Statistics and computing*, **17**:395–416, 2007.

[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.

[MBC18] Ekaterina Merkurjev, Andrea L Bertozzi, and Fan Chung. "A semi-supervised heat kernel pagerank MBO algorithm for data classification." *Communications in mathematical sciences*, **16**(5):1241–1265, 2018.

[MBG12] Jennifer R Marlon, Patrick J Bartlein, Daniel G Gavin, Colin J Long, R Scott Anderson, Christy E Briles, Kendrick J Brown, Daniele Colombaroli, Douglas J Hallett, Mitchell J Power, et al. "Long-term perspective on wildfires in the western USA." *Proceedings of the National Academy of Sciences*, **109**(9):E535–E543, 2012.

[MBY17] Ekaterina Merkurjev, Andrea Bertozzi, Xiaoran Yan, and Kristina Lerman. "Modified Cheeger and ratio cut methods using the Ginzburg–Landau functional for classification of high-dimensional data." *Inverse Problems*, **33**(7):074003, 2017.

[MBZ18] Lichao Mou, Lorenzo Bruzzone, and Xiao Xiang Zhu. "Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(2):924–935, 2018.

[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.

[MGC21] Alessandro Cesare Mondini, Fausto Guzzetti, Kang-Tsung Chang, Oriol Monserrat, Tapas Ranjan Martha, and Andrea Manconi. "Landslide failures detection and mapping using Synthetic Aperture Radar: Past, present and future." *Earth-Science Reviews*, **216**:103574, 2021.

[MGS13] Yifei Ma, Roman Garnett, and Jeff Schneider. "σ-optimality for active learning on gaussian random fields." *Advances in Neural Information Processing Systems*, **26**, 2013.

[MKB13] Ekaterina Merkurjev, Tijana Kostić, and Andrea L. Bertozzi. "An MBO Scheme on Graphs for Classification and Image Processing." *SIAM Journal on Imaging Sciences*, **6**(4):1903–1930, January 2013.

[MLN21] Oscar Manas, Alexandre Lacoste, Xavier Giró-i Nieto, David Vazquez, and Pau Rodriguez. "Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9414–9423, 2021.

[MLS22] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. "Is Homophily a Necessity for Graph Neural Networks?" In *International Conference on Learning Representations*, 2022.

[MM18] 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, 2018.

[MMS17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083*, 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.

[MNW23] Ekaterina Merkurjev, Duc Duy Nguyen, and Guo-Wei Wei. "Multiscale Laplacian learning." *Applied Intelligence*, **53**(12):15727–15746, 2023.

[MOS15] Juan J Manfredi, Adam M Oberman, and Alexander P Sviridov. "Nonlinear elliptic partial differential equations and p-harmonic functions on graphs." *Differential Integral Equations*, **28**(1-2):79–102, 2015.

[MSD19] Charlie Marshak, Marc Simard, and Michael Denbina. "Monitoring forest loss in ALOS/PALSAR time-series with superpixels." *Remote Sensing*, **11**(5):556, 2019.

[MSR19] Alessandro C Mondini, Michele Santangelo, Margherita Rocchetti, Enrica Rossetto, Andrea Manconi, and Oriol Monserrat. "Sentinel-1 SAR amplitude imagery for rapid landslide detection." *Remote sensing*, **11**(7):760, 2019.

[MT92] William B Meyer and Billie L Turner. "Human population growth and global land-use/cover change." *Annual review of ecology and systematics*, pp. 39–61, 1992.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[NAS24] NASA Earth Observatory. "Inferno Scars Valparaíso." `https://earthobservatory.nasa.gov/images/152417/inferno-scars-valparaiso`, 2024. [Online; accessed 16-Oct-2024].

[NBM22] Lorenzo Nava, Kushanav Bhuyan, Sansar Raj Meena, Oriol Monserrat, and Filippo Catani. "Rapid Mapping of Landslides on SAR Data by Attention U-Net." *Remote Sensing*, **14**(6), 2022.

[NIS18] NISAR Science Team. *NASA-ISRO SAR (NISAR) Mission Science Users' Handbook*. NASA Jet Propulsion Laboratory, 2018. 261 pp.

[NIS22] NISAR Science Algorithms Team. "Forest Disturbance Product Generation." `https://gitlab.com/nisar-science-algorithms/ecosystems/forest_disturbance/ProductGeneration`, 2022. Accessed: 2024-10-11.

[NJW01] Andrew Ng, Michael Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm." In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[NMB24] Lorenzo Nava, Alessandro Cesare Mondini, Kushanav Bhuyan, Chengyong Fang, Oriol Monserrat, Alessandro Novellino, and Filippo Catani. "Sentinel-1 SAR-based Globally Distributed Landslide Detection by Deep Neural Networks." *EarthArXiv*, 2024.

[NMC21] Lorenzo Nava, Oriol Monserrat, and Filippo Catani. "Improving landslide detection on SAR data through deep learning." *IEEE Geoscience and Remote Sensing Letters*, **19**:1–5, 2021.

[NSZ09] Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. "Semi-Supervised Learning with the Graph Laplacian: The Limit of Infinite Unlabelled Data." In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, p. 1330–1338, Red Hook, NY, USA, 2009. Curran Associates Inc.

[OA24] OpenAI, Josh Achiam, et al. "GPT-4 Technical Report.", 2024.

[Oll10] Yann Ollivier. "A survey of Ricci curvature for metric spaces and Markov chains." In *Probabilistic approach to geometry*, volume 57, pp. 343–382. Mathematical Society of Japan, 2010.

[OPE23] OPERA. "OPERA Level-2 Radiometric Terrain Corrected from Sentinel-1." Fairbanks, Alaska, USA: NASA Alaska Satellite Facility Distributed Active Archive Center, 2023. [Online; accessed 16-Oct-2024].

[OPE24a] OPERA Cal/Val Team. "dist-s1-events – GitHub repository." `https://github.com/OPERA-Cal-Val/dist-s1-events`, 2024. [Online; accessed 16-Oct-2024].

[OPE24b] OPERA Cal/Val Team. "RTC Sentinel-1 Table (JSON)." `https://github.com/OPERA-Cal-Val/dist-s1-validation-harness/blob/dev/data/rtc_s1_table.json.zip`, 2024. [Online; accessed 16-Oct-2024].

[OQ04] Chris Oliver and Shaun Quegan. *Understanding synthetic aperture radar images*. SciTech Publishing, 2004.

[PBC] Planet Labs PBC. "Planet Application Program Interface: In Space for Life on Earth.", 2018–.

[Pel04] W Richard Peltier. "Global glacial isostasy and the surface of the ice-age Earth: the ICE-5G (VM2) model and GRACE." *Annu. Rev. Earth Planet. Sci.*, **32**(1):111–149, 2004.

[PHV21] Bo Peng, Qunying Huang, Jamp Vongkusolkit, Song Gao, Daniel B. Wright, Zheng N. Fang, and Yi Qiang. "Urban Flood Mapping With Bitemporal Multispectral Imagery Via a Self-Supervised Learning Framework." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **14**:2001–2016, 2021.

[Pic24] Rob Picheta. "Satellite imagery shows scale of devastation after Papua New Guinea landslide." https://www.cnn.com/2024/05/28/asia/papua-new-guinea-landslide-satellite-imagery-intl/index.html, 2024.

[PLK24] Stefan Peters, Jixue Liu, Gunnar Keppel, Anna Wendleder, and Peiliang Xu. "Detecting coseismic landslides in GEE using machine learning algorithms on combined optical and radar imagery." *Remote Sensing*, **16**(10):1722, 2024.

[PO24] Amy Pickens and OPERA Cal/Val Team. "DIST Validation Repository." https://github.com/OPERA-Cal-Val/DIST-Validation/tree/main, 2024. Accessed: 2024-10-11.

[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.

[RAK20a] Clément Rambour, Nicolas Audebert, E Koeniguer, Bertrand Le Saux, M Crucianu, and Mihai Datcu. "Flood detection in time series of optical and sar images." *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **43**(B2):1343–1346, 2020.

[RAK20b] Clément Rambour, Nicolas Audebert, Elise Koeniguer, Bertrand Le Saux, Michel Crucianu, and Mihai Datcu. "SEN12-FLOOD: a SAR and Multispectral Dataset for Flood Detection.", 2020.

[RBE15] Robert A Rose, Dirck Byler, J Ron Eastman, Erica Fleishman, Gary Geller, Scott Goetz, Liane Guild, Healy Hamilton, Matt Hansen, Rachel Headley, et al. "Ten ways remote sensing can contribute to conservation." *Conservation Biology*, **29**(2):350–359, 2015.

[RGA18] Paul A Rosen, Eric M Gurrola, Piyush Agram, Joshua Cohen, Marco Lavalle, Bryan V Riel, Heresh Fattahi, Michael AG Aivazis, Mark Simons, and Sean M Buckley. "The InSAR scientific computing environment 3.0: a flexible framework for NISAR operational and user-led science processing." In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 4897–4900. IEEE, 2018.

[RGL23] Colorado J Reed, Ritwik Gupta, Shufan Li, Sarah Brockman, Christopher Funk, Brian Clipp, Kurt Keutzer, Salvatore Candido, Matt Uyttendaele, and Trevor Darrell. "Scale-mae: A scale-aware masked autoencoder for multiscale geospatial representation learning." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4088–4099, 2023.

[RHJ00] Paul A Rosen, Scott Hensley, Ian R Joughin, Fuk K Li, Soren N Madsen, Ernesto Rodriguez, and Richard M Goldstein. "Synthetic aperture radar interferometry." *Proceedings of the IEEE*, **88**(3):333–382, 2000.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986." *Biometrika*, **71**(599-607):6, 1986.

[RMS21] Johannes Reiche, Adugna Mullissa, Bart Slagter, Yaqing Gou, Nandin-Erdene Tsendbazar, Christelle Odongo-Braun, Andreas Vollrath, Mikaela J Weisse, Fred Stolle, Amy Pickens, et al. "Forest disturbance alerts for the Congo Basin using Sentinel-1." *Environmental Research Letters*, **16**(2):024005, 2021.

[ROF92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms." *Physica D: nonlinear phenomena*, **60**(1-4):259–268, 1992.

[RV93] Eric JM Rignot and Jakob J Van Zyl. "Change detection techniques for ERS-1 SAR data." *IEEE Transactions on Geoscience and Remote sensing*, **31**(4):896–906, 1993.

[RVV18] Johannes Reiche, Rob Verhoeven, Jan Verbesselt, Eliakim Hamunyela, Niels Wielaard, and Martin Herold. "Characterizing tropical forest cover loss using dense Sentinel-1 data and active fire alerts." *Remote Sensing*, **10**(5):777, 2018.

[SAS22] Philip Sellars, Angelica I. Aviles-Rivero, and Carola-Bibiane Schönlieb. "LaplaceNet: A Hybrid Graph-Energy Neural Network for Deep Semisupervised Classification." *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022.

[Set09] Burr Settles. "Active learning literature survey." *University of Wisconsin-Madison Department of Computer Sciences*, 2009.

[SFM23] Gustavo HX Shiroma, Heresh Fattahi, Franz Meyer, Seongsu Jeong, Luca Cinquini, Scott Collins, Bruce Chapman, Steven K Chan, Alexander L Handwerger, and David Bekaert. "The Opera Radiometric Terrain Corrected Sar Backscatter from Sentinel-1 (RTC-S1) Product." In *IGARSS 2023-2023 IEEE International Geoscience and Remote Sensing Symposium*, pp. 880–883. IEEE, 2023.

[SKL24] Anam Sabir, Unmesh Khati, Marco Lavalle, and Hari Shanker Srivastava. "Adapting CuSUM Algorithm for Site-Specific Forest Conditions to Detect Tropical Deforestation." *Remote Sensing*, **16**(20):3871, 2024.

[SKZ21] Oliver L Stephenson, Tobias Köhne, Eric Zhan, Brent E Cahill, Sang-Ho Yun, Zachary E Ross, and Mark Simons. "Deep learning-based damage mapping with InSAR coherence time series." *IEEE Transactions on Geoscience and Remote Sensing*, **60**:1–17, 2021.

[SLC24] Adam Stewart, Nils Lehmann, Isaac Corley, Yi Wang, Yi-Chia Chang, Nassim Ait Ait Ali Braham, Shradha Sehgal, Caleb Robinson, and Arindam Banerjee. "Ssl4eo-l: Datasets and foundation models for landsat imagery." *Advances in Neural Information Processing Systems*, **36**, 2024.

[Sma11] David Small. "Flattening Gamma: Radiometric Terrain Correction for SAR Imagery." *IEEE Transactions on Geoscience and Remote Sensing*, **49**(8):3081–3093, 2011.

[SOZ17] Zuoqiang Shi, Stanley Osher, and Wei Zhu. "Weighted nonlocal laplacian on interpolation from sparse data." *Journal of Scientific Computing*, **73**:1164–1177, 2017.

[ST19] Dejan Slepcev and Matthew Thorpe. "Analysis of p-Laplacian regularization in semisupervised learning." *SIAM Journal on Mathematical Analysis*, **51**(3):2085–2120, 2019.

[Sut14] I Sutskever. "Sequence to Sequence Learning with Neural Networks." *arXiv preprint arXiv:1409.3215*, 2014.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*, 2014.

[SZS13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199*, 2013.

[TCZ23] Gabriel Tseng, Ruben Cartuyvels, Ivan Zvonkov, Mirali Purohit, David Rolnick, and Hannah Kerner. "Lightweight, pre-trained transformers for remote sensing timeseries." *arXiv preprint arXiv:2304.14065*, 2023.

[TGC22] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature." In *International Conference on Learning Representations*, 2022.

[TLW25] Yu Tian, Zachary Lubberts, and Melanie Weber. "Curvature-based clustering on graphs." *Journal of Machine Learning Research*, **26**(52):1–67, 2025.

[TSW22] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training." *Advances in neural information processing systems*, **35**:10078–10093, 2022.

[UN24] UN. "Papua New Guinea landslide: 670 feared dead, says UN migration agency." `https://news.un.org/en/story/2024/05/1150246`, 2024.

[UNI24] UNICEF. "Bangladesh Situation Report No. 3 (Cyclone Remal and Floods), 11 September 2024." https://reliefweb.int/report/bangladesh/unicef-bangladesh-situation-report-no-3-cyclone-remal-and-floods-11-september-2024, 2024.

[Vas17] A Vaswani. "Attention is all you need." *Advances in Neural Information Processing Systems*, 2017.

[VCC17] Petar Velicković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks." *arXiv preprint arXiv:1710.10903*, 2017.

[VMB23] Karthik Venkataramani, Charles Z Marshak, David Bekaert, Marc Simard, Michael Denbina, Alexander L Handwerger, and Steven Chan. "Harmonizing SAR and Optical Data to Map Surface Water Extent: A Deep Learning Approach." In *IGARSS 2023-2023 IEEE International Geoscience and Remote Sensing Symposium*, pp. 3349–3352. IEEE, 2023.

[WBT25a] Adrien Weihs, Andrea L Bertozzi, and Matthew Thorpe. "Analysis of Semi-Supervised Learning on Hypergraphs." *arXiv preprint arXiv:2510.25354*, 2025.

[WBT25b] Adrien Weihs, Andrea L Bertozzi, and Matthew Thorpe. "Higher-order regularization learning on hypergraphs." *arXiv preprint arXiv:2510.26533*, 2025.

[WGY24] Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. "Heterophily-aware graph attention network." *Pattern Recognition*, **156**:110738, 2024.

[WO21] Bao Wang and Stan J Osher. "Graph interpolating activation improves both natural and robust accuracies in data-efficient deep learning." *European Journal of Applied Mathematics*, **32**(3):540–569, 2021.

[Woo17] Iain H Woodhouse. *Introduction to microwave remote sensing*. CRC press, 2017.

[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.

[WTT13] Bo Wang, Zhuowen Tu, and John K Tsotsos. "Dynamic label propagation for semi-supervised multi-class multi-label classification." In *Proceedings of the IEEE international conference on computer vision*, pp. 425–432, 2013.

[WZD23] Yifei Wang, Qi Zhang, Tianqi Du, Jiansheng Yang, Zhouchen Lin, and Yisen Wang. "A Message Passing Perspective on Learning Dynamics of Contrastive Learning." In *International Conference on Learning Representations*, 2023.

[WZL22] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. "Nodeformer: A scalable graph structure learning transformer for node classification." *Advances in Neural Information Processing Systems*, **35**:27387–27401, 2022.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747*, 2017.

[Yan10] Christopher J.C. Burges Yann LeCun, Corinna Cortes. "MNIST handwritten digit database." `http://yann.lecun.com/exdb/mnist/`, 2010.

[YCC21] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. "Florence: A new foundation model for computer vision." *arXiv preprint arXiv:2111.11432*, 2021.

[YHO15] Sang-Ho Yun, Kenneth Hudnut, Susan Owen, Frank Webb, Mark Simons, Patrizia Sacco, Eric Gurrola, Gerald Manipon, Cunren Liang, Eric Fielding, et al. "Rapid damage mapping for the 2015 M w 7.8 Gorkha earthquake using synthetic aperture radar data from COSMO–SkyMed and ALOS-2 Satellites." *Seismological Research Letters*, **86**(6):1549–1556, 2015.

[ZB11] Xueyuan Zhou and Mikhail Belkin. "Semi-supervised learning by higher order regularization." In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 892–900. JMLR Workshop and Conference Proceedings, 2011.

[ZDD19] Weiying Zhao, Charles-Alban Deledalle, Loïc Denis, Henri Maître, Jean-Marie Nicolas, and Florence Tupin. "Ratio-based Multitemporal SAR Images Denoising: RABASAR." *IEEE Transactions on Geoscience and Remote Sensing*, **57**(6):3552–3565, 2019.

[ZGG22] Yi Zheng, Rushin H Gindra, Emily J Green, Eric J Burks, Margrit Betke, Jennifer E Beane, and Vijaya B Kolachalama. "A graph-transformer for whole slide image classification." *IEEE transactions on medical imaging*, **41**(11):3003–3015, 2022.

[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.

[ZLL23] Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. "Coverage-centric Coreset Selection for High Pruning Rates." In *The Eleventh International Conference on Learning Representations*, 2023.

[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.

[ZZC16] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising." *CoRR*, **abs/1608.03981**, 2016.